SPECTRAL REGRESSION: A REGRESSION FRAMEWORK FOR EFFICIENT REGULARIZED SUBSPACE LEARNING

BY

DENG CAI

B.Eng., Tsinghua University, 2000
M.Eng., Tsinghua University, 2003

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Thomas S. Huang
Associate Professor Chengxiang Zhai
Associate Professor Kevin Chang

Abstract

Spectral methods have recently emerged as a powerful tool for dimensionality reduction and manifold learning. These methods use information contained in the eigenvectors of a data affinity (*i.e.*, item-item similarity) matrix to reveal the low dimensional structure in the high dimensional data. The most popular manifold learning algorithms include Locally Linear Embedding, ISOMAP, and Laplacian Eigenmap. However, these algorithms only provide the embedding results of training samples. There are many extensions of these approaches which try to solve the out-of-sample extension problem by seeking an embedding function in reproducing kernel Hilbert space. However, a disadvantage of all these approaches is that their computations usually involve eigen-decomposition of dense matrices which is expensive in both time and memory. In this thesis, we introduce a novel dimensionality reduction framework, called **Spectral Regression** (SR). SR casts the problem of learning an embedding function into a regression framework, which avoids eigen-decomposition of dense matrices. Also, with the regression as a building block, different kinds of regularizers can be naturally incorporated into our framework which makes it more flexible. SR can be performed in supervised, unsupervised and semi-supervised situation. It can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. We have applied our algorithms to several real world applications, e.g. face analysis, document representation and content-based image retrieval.

To my Parents.

Acknowledgements

I would like to express my greatest thanks to my advisor, Professor Jiawei Han, for his continued guidance and support during my graduate study.

I thank Xuanhui Wang, Qiaozhu Mei, Zheng Shao, Xifeng Yan and Yuxiao Hu for enjoyable collaborations over the years. I hope we keep working together. I am especially grateful to Xiaofei He for many discussions and insightful suggestions.

Table of Contents

List of Tables

List of Figures

List of Symbols

n	The number of sample points.
m	The number of features.
c	The number of classes.
$\mathbf{x}_i$	The i-th sample point.
X	The data matrix.
$\bar{X}$	The centered data matrix.
S_b	The between-class scatter matrix.
S_w	The within-class scatter matrix.
S_t	The total scatter matrix.
W	The affinity matrix.
L	The graph Laplacian matrix.

Chapter 1

Introduction

Dimensionality reduction has been a key problem in many fields of information processing, such as machine learning, data mining, information retrieval, and pattern recognition. Practical algorithms in supervised machine learning degrade in performance (prediction accuracy) when faced with many features that are not necessary for predicting the desired output. An important question in the fields of machine learning, knowledge discovery, computer vision and pattern recognition is how to extract a small number of good features. A common way to attempt to resolve this problem is to use dimensionality reduction techniques.

One of the most popular dimensionality reduction algorithms might be Principal Component Analysis (PCA) [57]. PCA performs dimensionality reduction by projecting the original m-dimensional data onto the $d(\ll m)$-dimensional linear subspace spanned by the leading eigenvectors of the data's covariance matrix. Its goal is to find a set of mutually orthogonal basis functions that capture the directions of maximum variance in the data so that the pairwise *Euclidean* distances can be best preserved. If the data is embedded in a linear subspace, PCA is guaranteed to discover the dimensionality of the subspace and produces a compact representation.

In many real world problems, however, there is no evidence that the data is sampled from a linear subspace. For example, it is always believed that the face images are sampled from a nonlinear low-dimensional manifold which is embedded in the high-dimensional ambient space [47]. This motivates us to consider manifold based techniques for dimensionality reduction. Recently, various manifold learning techniques, such as ISOMAP [75],

Locally Linear Embedding (LLE) [68] and Laplacian Eigenmap [4] have been proposed which reduce the dimensionality of a *fixed* training set in a way that maximally preserve certain inter-point relationships. LLE and Laplacian Eigenmap are local methods which attempt to preserve local geometry of the data; essentially, they seek to map nearby points on the manifold to nearby points in the low-dimensional representation. ISOMAP is a global method which attempts to preserve geometry at all scales, mapping nearby points on the manifold to nearby points in low-dimensional space, and faraway points to faraway points. One of the major limitations of these methods is that they do not generally provide a functional mapping between the high and low dimensional spaces that are valid both on and off the training data.

There are a lot of approaches that try to address this issue by explicitly requiring an embedding function either linear or in reproducing kernel Hilbert space (RKHS) when minimizing the objective function [46, 7, 84]. They provide natural out-of-sample extensions of Lapalcian Eigenmaps, LLE and Isomap. However, the computation of these methods involves eigen-decomposition of dense matrices which is expensive in both time and memory. It is almost infeasible to apply these approaches on large data sets. Some other approaches address this issue through a kernel view of LLE, Isomap and Laplacian Eigenmaps [6, 39]. They interpret these spectral embedding algorithms as learning the principal eigenfunctions of an operator defined from a kernel and the unknown data generating density. Such kernel is usually data dependant[1]. To obtain the embedding result of an unseen example, we need to calculate the kernel function values of this unseen example with all the training samples which may not be possible in some situations.

In this thesis, we propose a novel dimensionality reduction algorithm, called *Spectral Regression* (SR). The proposed algorithm is fundamentally based on regression and spectral graph analysis [26]. It can be performed either in supervised, unsupervised or semi-supervised situations. Specifically, we first construct an affinity graph over both labeled

[1]The kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ depends not only on $\mathbf{x}_i$ and $\mathbf{x}_j$ but also on the whole data set.

and unlabeled points to discover the intrinsic discriminant structure in the data. This graph is used to learn responses for both labeled and unlabeled points. Once the responses are obtained, the ordinary regression is then applied for learning the embedding function.

The points below highlight several aspects of our approach:

1. SR casts the problem of learning an embedding function into a regression framework, which avoids eigen-decomposition of dense matrices. With different graph matrix W, SR provides the efficient solutions of Linear Discriminant Analysis (LDA) [19], Locality Preserving Projection (LPP) [46, 17, 20], Neighborhood Preserving Embedding (NPE)[45, 17], Isometrix Projeciton (IsoP)[13], Locality Sensitive Discriminant Analysis (LSDA) [21] and much more.

2. With regression as the building block, various kinds of regularization techniques can be easily incorporated in SR which makes it more flexible (e.g., L_1-norm regularizer to produce sparse projections [14]).

3. SR can be performed in supervised [19], unsupervised [20] and semi-supervised [16, 15] situations. It can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data.

4. SR may be conducted in the original space or in the reproducing kernel Hilbert space (RKHS) into which data points are mapped. This gives rise to kernel SR (efficent solutions for many kernel subspace learning algorithms [12]).

Chapter 2

Graph Embedding View of Subspace Learning

Let $\mathbf{x}_1, \cdots, \mathbf{x}_n$ be the n data points sampled from an underlying submanifold $\mathcal{M}$ embedded in $\mathbb{R}^m$, dimensionality reduction (or, subspace learning) aims at finding $\{\mathbf{z}_i\}_{i=1}^n \subset \mathbb{R}^d, d \ll m$, where $\mathbf{z}_i$ can "represent" $\mathbf{x}_i$. In the past decades, many algorithms, either supervised or unsupervised, have been proposed to solve this problem. Despite the different motivations of these algorithms, they can be nicely interpreted in a general *graph embedding* framework. In this chapter, we give a detailed analysis of this framework and its linear extension.

2.1 Manifold Learning and Graph Embedding

We begin with a brief review of Locally Linear Embedding (LLE) [68], Isomap [75], and Laplacian Eigenmaps [4], three of the most popular manifold learning techniques. We then discuss how these three algorithms can be unified in a graph embedding framework with different graphs. For simplicity, we consider one dimensional mapping. Let y_i be the one dimensional map of $\mathbf{x}_i$, $i = 1, \cdots, n$.

2.1.1 Locally Linear Embedding

The basic idea of LLE is that the data points might reside on a nonlinear submanifold, but it might be reasonable to assume that each local neighborhood is linear. Thus, we can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. Specifically, we first construct a k nearest neighbor graph G

with weight matrix M. Reconstructing errors are measured by the cost function [68]:

$$\phi(M) = \sum_{i=1}^{n} \|\mathbf{x}_i - \sum_{j=1}^{n} M_{ij}\mathbf{x}_j\|^2, \quad s.t. \ \sum_{j=1}^{n} M_{ij} = 1$$

which adds up the squared distances between all the data points and their reconstructions. Note that, M_{ij} vanishes for distant data points. Please see [68] for how to find a M which minimizes $\phi(M)$. Consider the problem of mapping the original data points to a line so that each data point on the line can be represented as a linear combination of its neighbors with the coefficients M_{ij}. Let $\mathbf{y} = (y_1, y_2, \cdots, y_m)^T$ be such a map. A reasonable criterion for choosing a "good" map is to minimize the following loss function [68]:

$$\Phi(\mathbf{y}) = \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{n} M_{ij} y_j\Big)^2$$

This loss function, like the previous one, is based on locally linear reconstruction errors, but here we fix the weights M_{ij} while optimizing the coordinates y_i. It can be shown that the optimal embedding $\mathbf{y}$ is given by the eigenvector corresponding to the **minimum** eigenvalue of the following eigen-problem:

$$(I - M)^T(I - M)\mathbf{y} = \lambda\mathbf{y} \tag{2.1}$$

where I is an $m \times m$ identity matrix.

Define matrix $W_{LLE} = M + M^T - M^T M$, we can rewrite the eigen-problem in Eqn. (2.1) as

$$\begin{aligned} & (I - W_{LLE})\mathbf{y} = \lambda\mathbf{y} \\ \Rightarrow \quad & W_{LLE}\mathbf{y} = (1 - \lambda)\mathbf{y} \end{aligned}$$

Thus, the optimal embedding $\mathbf{y}$ is given by the eigenvector corresponding to the **maximum**

eigenvalue of the eigen-problem:

$$W_{LLE}\mathbf{y} = \lambda\mathbf{y} \tag{2.2}$$

2.1.2 ISOMAP

Let $d_{\mathcal{M}}$ be the geodesic distance measure on $\mathcal{M}$ and d the standard Euclidean distance measure in $\mathbb{R}^m$. ISOMAP aims to find a Euclidean embedding such that Euclidean distances in $\mathbb{R}^m$ can provide a good approximation to the geodesic distances on $\mathcal{M}$. That is,

$$f^{opt} = \arg\min_{f} \sum_{i,j} \Big(d_{\mathcal{M}}(\mathbf{x}_i, \mathbf{x}_j) - d\big(f(\mathbf{x}_i), f(\mathbf{x}_j)\big) \Big)^2 \tag{2.3}$$

In real life data set, the underlying manifold $\mathcal{M}$ is often unknown and hence the geodesic distance measure is also unknown. In order to discover the intrinsic geometrical structure of $\mathcal{M}$, we first construct a k nearest neighbor graph G over all data points to model the local geometry. Once the graph is constructed, the geodesic distances $d_{\mathcal{M}}(i,j)$ between all pairs of points on the manifold $\mathcal{M}$ can be estimated by computing their shortest path distances $d_G(i,j)$ on the graph G. The procedure is as follows: initialize $d_G(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_i, \mathbf{x}_j)$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are linked by an edge; $d_G(\mathbf{x}_i, \mathbf{x}_j) = \infty$ otherwise. Then for each value of $l = 1, 2, \cdots, n$ in turn, replace all entries $d_G(\mathbf{x}_i, \mathbf{x}_j)$ by $\min\{d_G(\mathbf{x}_i, \mathbf{x}_j), d_G(\mathbf{x}_i, \mathbf{x}_l) + d_G(\mathbf{x}_l, \mathbf{x}_j)\}$. The matrix of final values $D_G = \{d_G(\mathbf{x}_i, \mathbf{x}_j)\}$ will contain the shortest path distances between all pairs of points in G. This procedure is named Floyd-Warshall algorithm [27]. More efficient algorithms exploiting the sparse structure of the neighborhood graph can be found in [37]. Let D_Y denote the matrix of Euclidean distances in the reduced subspace, i.e. $\{d_Y(i,j) = \|y_i - y_j\|\}$. Thus, ISOMAP aims to minimize the cost function:

$$\|\tau(D_G) - \tau(D_Y)\|_{L^2}$$

where the τ operator converts distances to inner products, which uniquely characterize

the geometry of the data in a form that supports efficient optimization [75]. Specifically, $\tau(D) = -HSH/2$, where $S_{ij} = D_{ij}^2$ and $H = I - \frac{1}{m}\mathbf{e}\mathbf{e}^T$, $\mathbf{e} = (1, 1, \cdots, 1)^T$. Define $W_{Isomap} = \tau(D_G)$, it can be shown that the optimal embedding $\mathbf{y} = (y_1, \cdots, y_m)$ is given by the eigenvector of the matrix W_{Isomap} corresponding to the largest eigenvalue.

$$W_{ISOMAP}\mathbf{y} = \lambda\mathbf{y} \tag{2.4}$$

2.1.3 Laplacian Eigenmap

Laplacian Eigenmap is based on spectral graph theory [26]. Given a p nearest neighbor graph G with weight matrix W, which can be defined as follows:

$$W_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in N_p(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_p(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

where $N_p(\mathbf{x}_i)$ denotes the set of p nearest neighbors of $\mathbf{x}_i$.

The optimal maps can be obtained by solving the following minimization problem:

$$\min_{\mathbf{y}} \sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij} = \min_{\mathbf{y}} 2\mathbf{y}^T L\mathbf{y}$$

where $L = D - W$ is the *graph Laplacian* [26] and $D_{ii} = \sum_j W_{ij}$. The objective function with our choice of weights W_{ij} incurs a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped far apart. Therefore, minimizing it is an attempt to ensure that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are "close" then y_i and y_j are close as well.

The optimal embedding $\mathbf{y}$ is given by the eigenvector corresponding to the **minimum** eigenvalue of the following generalized eigen-problem

$$L\mathbf{y} = \lambda D\mathbf{y}, \tag{2.6}$$

which is equivalent to find the eigenvector corresponding to the **maximum** eigenvalue of the following generalized eigen-problem

$$W\mathbf{y} = \lambda D\mathbf{y}, \tag{2.7}$$

2.1.4 Graph Embedding

All the above three manifold learning algorithms encode the intrinsic structure information of the data in a graph weight matrix. And all the three optimization problems end up with the similar eigen-problems.

In the following, we consider the general graph embedding problem. Given a graph G with n vertices, each representing a data point, let W be a symmetric $n \times n$ matrix with W_{ij} having the weight of the edge joining vertices i and j. The G and W can be defined to characterize certain statistical or geometric properties of the data set. The purpose of graph embedding is to represent each vertex of the graph as a low dimensional vector that preserves similarities between the vertex pairs, where similarity is measured by the edge weight.

Let $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$ be the map from the graph vertices to the real line. The optimal **y** tries to minimize

$$\sum_{i,j}(y_i - y_j)^2 W_{ij}$$

under appropriate constraint. This objective function incurs a heavy penalty if neighboring vertices i and j (with a large W_{ij}) are mapped far apart. Therefore, minimizing it is an attempt to ensure that if vertices i and j are "close" then y_i and y_j are close as well [38]. With some simple algebraic formulations, we have

$$\sum_{i,j}(y_i - y_j)^2 W_{ij} = 2\mathbf{y}^T L\mathbf{y},$$

where L is the *graph Laplacian* as discussed before. Finally, the minimization problem reduces to find

$$\mathbf{y}^* = \arg\min_{\mathbf{y}^T D\mathbf{y}=1} \mathbf{y}^T L\mathbf{y} = \arg\min \frac{\mathbf{y}^T L\mathbf{y}}{\mathbf{y}^T D\mathbf{y}} = \arg\max \frac{\mathbf{y}^T W\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}, \tag{2.8}$$

where the constraint $\mathbf{y}^T D\mathbf{y} = 1$ removes an arbitrary scaling factor in the embedding.

It is clear that the three manifold learning algorithms we discussed before can be interpreted in this framework with different choices of W and D. The two matrices W and D play the essential role in this graph embedding approach. The choices of these two graph matrices can be very flexible. In later discussion, we use GE(W, D) to denote the graph embedding with maximization problem of $\max(\mathbf{y}^T W\mathbf{y})/(\mathbf{y}^T D\mathbf{y})$.

All the above mentioned manifold learning algorithms are nonlinear. They are defined only on the training data points and therefore can not be directly applied to supervised learning problems. To overcome this limitation, some methods for out-of-sample extension have been proposed [6]. Bengio *et al*. proposed a unified framework for extending LLE, Isomap, and Laplacian Eigenmap [6]. This framework is based on seeing these algorithms as learning eigenfunctions of a data-dependent kernel. The Nyström formula is used to obtain an embedding for a new data point. However, To obtain the embedding result of an unseen example, we need to calculate the kernel function values of this unseen example with all the training samples which may not be possible in some situations[1]. In the following sections, we will discuss how we can solve this issue by explicitly requiring an embedding function either linear or in the reproducing kernel Hilbert space (RKHS) when minimizing the objective function.

[1] *e.g*., the data dependant kernel is constructed by integrating label information. To calculate $K(\mathbf{x}_i, \mathbf{x}_j)$, we need to know whether $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same label. Since the label of an unseen example is usually unavailable, we can not calculate the kernel function values of this unseen example with all the training samples.

2.2 Linear Extension of Graph Embedding

If we choose a linear function, *i.e.*, $y_i = f(\mathbf{x}_i) = \mathbf{a}^T\mathbf{x}_i$, we have $\mathbf{y} = X^T\mathbf{a}$ where $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$. Eqn. (2.8) can be rewritten as:

$$\mathbf{a}^* = \arg\max \frac{\mathbf{y}^T W \mathbf{y}}{\mathbf{y}^T D \mathbf{y}} = \arg\max \frac{\mathbf{a}^T X W X^T \mathbf{a}}{\mathbf{a}^T X D X^T \mathbf{a}}. \tag{2.9}$$

The optimal **a**'s are the eigenvectors corresponding to the maximum eigenvalue of eigenproblem:

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}.$$

This approach is called linear extension of graph embedding. It can certainly be applied on LLE, Isomap and Laplacian Eigenmap which leads to Neighborhood Preserving Embedding (NPE) [45], Isometric Projection [13] and Locality Preserving Projection (LPP) [46]. In the following, we will discuss three other linear dimensionality reduction algorithms. These three algorithms are proposed with different motivations. However, we will see that all these algorithms can be formulated as linear extension of graph embedding with different W and D.

2.2.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) [35] is one of the most well known supervised dimensionality reduction algorithms. It seeks directions on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. Suppose we have a set of n samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \in \mathbb{R}^m$, belonging to c classes. The objective function of LDA is as follows:

$$\mathbf{a}_{opt} = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}}, \tag{2.10}$$

$$S_b = \sum_{k=1}^{c} n_k(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T, \tag{2.11}$$

$$S_w = \sum_{k=1}^{c} \left(\sum_{i=1}^{n_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})^T \right), \tag{2.12}$$

where $\boldsymbol{\mu}$ is the total sample mean vector, n_k is the number of samples in the k-th class, $\boldsymbol{\mu}^{(k)}$ is the average vector of the k-th class, and $\mathbf{x}_i^{(k)}$ is the i-th sample in the k-th class. We call S_w the within-class scatter matrix and S_b the between-class scatter matrix.

Define the total scatter matrix $S_t = \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$, we have $S_t = S_b + S_w$ [35]. The objective function of LDA in Eqn. (2.10) is equivalent to

$$\mathbf{a}_{opt} = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}}. \tag{2.13}$$

The optimal **a**'s are the eigenvectors corresponding to the non-zero eigenvalue of eigen-problem:

$$S_b \mathbf{a} = \lambda S_t \mathbf{a}. \tag{2.14}$$

Since the rank of S_b is bounded by $c-1$, there are at most $c-1$ eigenvectors corresponding to non-zero eigenvalues [35].

Without loss of generality, we assume $\boldsymbol{\mu} = \mathbf{0}$.[2] We have

$$\begin{aligned} S_b &= \sum_{k=1}^{c} n_k(\boldsymbol{\mu}^{(k)})(\boldsymbol{\mu}^{(k)})^T \\ &= \sum_{k=1}^{c} n_k \left(\frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i^{(k)} \right) \left(\frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i^{(k)} \right)^T \\ &= \sum_{k=1}^{c} X^{(k)} W^{(k)} (X^{(k)})^T \end{aligned} \tag{2.15}$$

where $W^{(k)}$ is a $n_k \times n_k$ matrix with all the elements equal to $1/n_k$ and $X^{(k)} = [\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_{n_k}^{(k)}]$ denote the data matrix of k-th class.

[2] This can be achieved by centering the data, *i.e.*, subtract the mean vector from all the sample vectors.

Let the data matrix $X = [X^{(1)}, \cdots, X^{(c)}]$ and define a $n \times n$ matrix W_{LDA} as:

$$W_{LDA} = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \tag{2.16}$$

We have

$$S_b = \sum_{k=1}^{c} X^{(k)} W^{(k)} (X^{(k)})^T = X W_{LDA} X^T. \tag{2.17}$$

Thus, the objective function of LDA in Eqn. (2.13) can be rewritten as

$$\mathbf{a}_{opt} = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}} = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T X W_{LDA} X^T \mathbf{a}}{\mathbf{a}^T X X^T \mathbf{a}}. \tag{2.18}$$

Thus, LDA can also be interpreted as a linear extension of graph embedding approach.

2.2.2 Semi-supervised Discriminant Analysis

In this subsection, we introduce a semi-supervised subspace learning algorithm, called Semi-supervised Discriminant Analysis (SDA), which can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. SDA is fundamentally developed from LDA and LPP.

LDA aims to find a projection vector $\mathbf{a}$ such that the ratio between $\mathbf{a}^T S_b \mathbf{a}$ and $\mathbf{a}^T S_t \mathbf{a}$ is maximized. When there is no sufficient training sample, overfitting may happen. A typical way to prevent overfitting is to impose a regularizer [41]. The optimization problem of the regularized version of LDA can be written as follows:

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a} + \alpha J(\mathbf{a})} \tag{2.19}$$

where $J(\mathbf{a})$ controls the learning complexity of the hypothesis family, and the coefficient

α controls balance between the model complexity and the empirical loss. One of the most popular regularizers is the Tikhonov regularizer [76]:

$$J(\mathbf{a}) = \|\mathbf{a}\|^2.$$

LDA model with Tikhonov regularizer is usually referred as Regularized Discriminant Analysis (RDA) [34].

The regularizer term $J(\mathbf{a})$ provides us the flexibility to incorporate our prior knowledge on some particular applications. When a set of unlabeled examples available, we aim to construct a $J(\mathbf{a})$ incorporating the manifold structure. The key to semi-supervised learning algorithm is the prior assumption of consistency. For classification, it means nearby points are likely to have the same label [87]. For dimensionality reduction, it can be interpreted as nearby points will have similar embeddings (low-dimensional representations). Given a set of examples $\{\mathbf{x}_i\}_{i=1}^N$, we can use a p-nearest neighbor graph G to model the relationship between nearby data points. The corresponding weight matrix W_N can be defined as in Eq. (2.5), where the subscript N denotes that W_N is with size $N \times N$.

In general, the mapping function should be as smooth as possible on the graph. Specifically, if two data points are linked by an edge, they are likely to be in the same class. Moreover, the data points lying on a densely linked subgraph are likely to have the same label. Thus, a natural regularizer can be defined as follows:

$$J(\mathbf{a}) = \sum_{ij} \left(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j\right)^2 W_{N,ij} \tag{2.20}$$

This formulation is motivated from spectral dimensionality reduction [4, 46], which also plays a key role in spectral clustering [62] and various kinds of graph based semi-supervised learning algorithms [5, 24, 71].

Let $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$. We have

$$\begin{aligned} J(\mathbf{a}) &= \sum_{ij} (\mathbf{a}^T \mathbf{x}_i - \mathbf{a}^T \mathbf{x}_j)^2 W_{N,ij} \\ &= 2 \sum_i \mathbf{a}^T \mathbf{x}_i D_{ii} \mathbf{x}_i^T \mathbf{a} - 2 \sum_{ij} \mathbf{a}^T \mathbf{x}_i W_{N,ij} \mathbf{x}_j^T \mathbf{a} \\ &= 2\mathbf{a}^T X (D - W_N) X^T \mathbf{a} \\ &= 2\mathbf{a}^T X L X^T \mathbf{a} \end{aligned}$$

where D is a diagonal matrix; its entries are column (or row, since W_N is symmetric) sum of S, $D_{ii} = \Sigma_j W_{N,ij}$. $L = D - W_N$ is the Laplacian matrix [26].

With this data dependent regularizer, we get the objective function of our semi-supervised discriminant analysis:

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T \big(S_t + \alpha X L X^T\big) \mathbf{a}}. \tag{2.21}$$

The projective vector **a** that maximizes the objective function is given by the maximum eigenvalue solution to the generalized eigenvalue problem:

$$S_b \mathbf{a} = \lambda (S_t + \alpha X L X^T) \mathbf{a} \tag{2.22}$$

Without loss of generality, we assume that the first n data points are labeled and they are ordered according to their labels. We use $X_n = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ to denote the labeled data matrix. Define the weight matrix $W_{SDA} \in \mathbb{R}^{N \times N}$ as

$$W_{SDA} = \begin{bmatrix} W_{LDA} & 0 \\ 0 & 0 \end{bmatrix}, \qquad \widetilde{I} = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$$

where $W_{LDA} \in \mathbb{R}^{n \times n}$ is defined in Eqn. (2.16) and I is an identity matrix of size $n \times n$.

Based on Eqn. (2.17), we have

$$S_b = X_n W_{LDA} X_n^T = X W_{SDA} X^T \tag{2.23}$$

and

$$S_t = X_n X_n^T = X \widetilde{I} X^T. \tag{2.24}$$

Thus, the objective function of SDA in eqn. (2.25) can be rewritten as

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T X W_{SDA} X^T \mathbf{a}}{\mathbf{a}^T X \big(\widetilde{I} + \alpha L\big) X^T \mathbf{a}}, \tag{2.25}$$

which again is linear extension of a graph embedding problem.

2.2.3 Locality Sensitive Discriminant Analysis

As we described previously, naturally occurring data may be generated by structured systems with possibly much fewer degrees of freedom than the ambient dimension would suggest. Thus we consider the case when the data lives on or close to a submanifold of the ambient space. One hopes then to estimate geometrical and discriminant properties of the submanifold from random points lying on this unknown submanifold. In this section, we consider the particular question of maximizing *local* margin between different classes.

Recall that we can use a p-nearest neighbor graph G with weight matrix W to characterize the local geometry of the data manifold. In order to discover both geometrical and discriminant structure of the data manifold, we construct two graphs, i.e. *within-class graph* G_w and *between-class graph* G_b. Let $l(\mathbf{x}_i)$ be the class label of $\mathbf{x}_i$. For each data point $\mathbf{x}_i$, the set $N(\mathbf{x}_i)$ can be naturally split into two subsets, $N_b(\mathbf{x}_i)$ and $N_w(\mathbf{x}_j)$. $N_w(\mathbf{x}_i)$ contains the neighbors sharing the same label with $\mathbf{x}_i$, while $N_b(\mathbf{x}_i)$ contains the neighbors

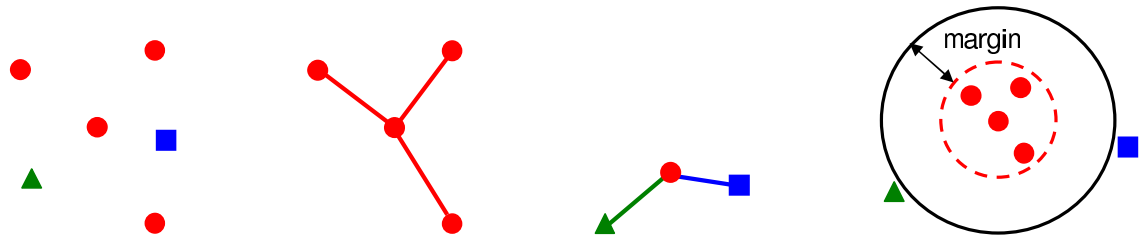


Figure 2.1: (a) The center point has five neighbors. The points with the same color and shape belong to the same class. (b) The *within-class graph* connects nearby points with the same label. (c) The *between-class graph* connects nearby points with different labels. (d) After Locality Sensitive Discriminant Analysis, the margin between different classes is maximized.

having different labels. Specifically,

$$N_w(\mathbf{x}_i) = \{\mathbf{x}_i^j | l(\mathbf{x}_i^j) = l(\mathbf{x}_i), 1 \leq j \leq p\}$$

$$N_b(\mathbf{x}_i) = \{\mathbf{x}_i^j | l(\mathbf{x}_i^j) \neq l(\mathbf{x}_i), 1 \leq j \leq p\}$$

Clearly, $N_b(\mathbf{x}_i) \cap N_w(\mathbf{x}_i) = \varnothing$ and $N_b(\mathbf{x}_i) \cup N_w(\mathbf{x}_i) = N(\mathbf{x}_i)$. Let W_w and W_b be the weight matrices of G_w and G_b, respectively. We define:

$$W_{b,ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in N_b(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_b(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{2.26}$$

$$W_{w,ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in N_w(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_w(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{2.27}$$

It is clear to see $W = W_b + W_w$ and the nearest neighbor graph G can be thought of as a combination of within-class graph G_w and between-class graph G_b.

Now consider the problem of mapping the within-class graph and between-class graph to a line so that connected points of G_w stay as close together as possible while connected points of G_b stay as distant as possible. Let $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$ be such a map. A

reasonable criterion for choosing a "good" map is to optimize the following two objective functions:

$$\min \sum_{ij} (y_i - y_j)^2 W_{w,ij} \tag{2.28}$$

$$\max \sum_{ij} (y_i - y_j)^2 W_{b,ij} \tag{2.29}$$

under appropriate constraints. The objective function (2.28) on within-class graph incurs a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped far apart while they are actually in the same class. Likewise, the objective function (2.29) on between-class graph incurs a heavy penalty if neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped close together while they actually belong to different classes. Therefore, minimizing (2.28) is an attempt to ensure that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close and sharing the same label then y_i and y_j are close as well. Also, maximizing (2.29) is an attempt to ensure that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close but have different labels then y_i and y_j are far apart. The learning procedure is illustrated in Figure 2.1.

Suppose $\mathbf{a}$ is a projection vector, that is, $\mathbf{y}^T = \mathbf{a}^T X$, where $X = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ is a $m \times n$ matrix. By simple algebra formulation, the objective function (2.28) can be reduced to

$$\begin{aligned}
& \frac{1}{2} \sum_{ij} (y_i - y_j)^2 W_{w,ij} \\
= \; & \frac{1}{2} \sum_{ij} \left(\mathbf{a}^T \mathbf{x}_i - \mathbf{a}^T \mathbf{x}_j \right)^2 W_{w,ij} \\
= \; & \sum_i \mathbf{a}^T \mathbf{x}_i D_{w,ii} \mathbf{x}_i^T \mathbf{a} - \sum_{ij} \mathbf{a}^T \mathbf{x}_i W_{w,ij} \mathbf{x}_j^T \mathbf{a} \\
= \; & \mathbf{a}^T X D_w X^T \mathbf{a} - \mathbf{a}^T X W_w X^T \mathbf{a}
\end{aligned}$$

where D_w is a diagonal matrix; its entries are column (or row, since W_w is symmetric) sum

of W_w, $D_{w,ii} = \sum_j W_{w,ij}$. Similarly, the objective function (2.29) can be reduced to

$$\begin{aligned} & \frac{1}{2}\sum_{ij}(y_i - y_j)^2 W_{b,ij} \\ = \; & \frac{1}{2}\sum_{ij}\left(\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\mathbf{x}_j\right)^2 W_{b,ij} \\ = \; & \mathbf{a}^T X(D_b - W_b)X^T\mathbf{a} \\ = \; & \mathbf{a}^T X L_b X^T\mathbf{a} \end{aligned}$$

where D_b is a diagonal matrix; its entries are column (or row, since W_b is symmetric) sum of W_b, $D_{b,ii} = \sum_j W_{b,ij}$. $L_b = D_b - W_b$ is the Laplacian matrix of G_b.

Note that, the matrix D_w provides a natural measure on the data points. If $D_{w,ii}$ is large, then it implies that the class containing $\mathbf{x}_i$ has a high density around $\mathbf{x}_i$. Therefore, the bigger the value of $D_{w,ii}$ is, the more "important" is $\mathbf{x}_i$. Therefore, we impose a constraint as follows:

$$\mathbf{y}^T D_w \mathbf{y} = 1 \Rightarrow \mathbf{a}^T X D_w X^T \mathbf{a} = 1$$

Thus, the objective function (2.28) becomes the following:

$$\min_{\mathbf{a}} \; 1 - \mathbf{a}^T X W_w X^T \mathbf{a} \tag{2.30}$$

or equivalently,

$$\max_{\mathbf{a}} \; \mathbf{a}^T X W_w X^T \mathbf{a} \tag{2.31}$$

And the objective function (2.29) can be rewritten as follows:

$$\max_{\mathbf{a}} \; \mathbf{a}^T X L_b X^T \mathbf{a} \tag{2.32}$$

Finally, the optimization problem reduces to finding:

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T X\big(\alpha L_b + (1-\alpha)W_w\big)X^T\mathbf{a}}{\mathbf{a}^T X D_w X^T \mathbf{a}}, \tag{2.33}$$

which again is linear extension of a graph embedding problem.

2.3 Computational and Complexity Analysis

The linear extension of graph embedding ends up with solving the generalized eigen-problem

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a}. \tag{2.34}$$

To get a stable solution of this eigen-problem, the matrices XDX^T is required to be non-singular [73] which is not true when the number of features is larger than the number of samples. There are two methods to solve this problem. The first one is by using Singular Value Decomposition (SVD) of X.

Suppose $rank(X) = r$, the SVD decomposition of X is

$$X = U\Sigma V^T \tag{2.35}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ and $\sigma_1 \geq \cdots \geq \sigma_r > 0$ are the singular values of X, $U \in \mathbb{R}^{m\times r}$, $V =\in \mathbb{R}^{n\times r}$ and $U^TU = V^TV = I$. Let $\widetilde{X} = U^TX = \Sigma V^T$ and $\mathbf{b} = \Sigma U^T\mathbf{a}$,

we have

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{2.36}$$

$$\Rightarrow \quad U\Sigma V^T WV\Sigma U^T\mathbf{a} = \lambda U\Sigma V^T DV\Sigma U^T\mathbf{a} \tag{2.37}$$

$$\Rightarrow \quad U\Sigma V^T WV\mathbf{b} = \lambda U\Sigma V^T DV\mathbf{b} \tag{2.38}$$

$$\Rightarrow \quad \Sigma^{-1}U^T U\Sigma V^T WV\mathbf{b} = \lambda\Sigma^{-1}U^T U\Sigma V^T DV\mathbf{b} \tag{2.39}$$

$$\Rightarrow \quad V^T WV\mathbf{b} = \lambda V^T DV\mathbf{b} \tag{2.40}$$

It is clear that $V^T DV$ is nonsingular and the eigen-problem in Eqn. (2.40) can be stably solved. After we get $\mathbf{b}^*$, the $\mathbf{a}^*$ can be obtained by

$$\mathbf{a}^* = U\Sigma^{-1}\mathbf{b}^*. \tag{2.41}$$

The above SVD approach has been widely used in many subspace learning algorithms (*e.g.*, LDA [18] and LPP [47]) to solve the singularity problem. For clarity, we name this approach as SVD+LGE (Linear Graph Embedding).

The second method is using the idea of regularization, by adding constant values to the diagonal elements of XDX^T, as $XDX^T+\gamma I$, for $\gamma > 0$. It is easy to see that $XDX^T+\gamma I$ is nonsingular. This method is used in [34].

2.3.1 Complexity Analysis of General Linear Graph Embedding

Now let us analyze the computational complexity of both SVD+LGE and the regularization approaches. We consider the case that the number of features (m) is larger than the number of samples (n) and use the term *flam* [72], a compound operation consisting of one addition and one multiplication, to present operation counts.

The most efficient algorithm to calculate the SVD decomposition requires $\frac{3}{2}n^2m + \frac{9}{2}n^3$ flam [73]. When $n < m$, the rank of X is usually of n. Thus, V is square matrix of

size $n \times n$. The calculation of matrices VWV^T and VDV^T requires at least $2n^3$ flam. The eigen-problem in Eqn. (2.40) requires $\frac{9}{2}n^3$ flam [73]. Overall, the time complexity of SVD+LGE approach measured by flam is

$$\frac{3}{2}n^2m + 11n^3,$$

which is cubic-time complexity with respect to n. For large scale high dimensional data, the SVD+LGE approach is unlikely to be applied.

In the regularization approach, The calculation of matrices XWX^T and $XDX^T + \gamma I$ requires at least $2nm^2$ flam. The generalized eigen-problem requires $\frac{9}{2}m^3$ flam. Overall, the time complexity of the regularization approach measured by flam is

$$2nm^2 + \frac{9}{2}m^3,$$

which is cubic-time complexity with respect to m. The regularization approach is also unlikely to be applied for large scale high dimensional data.

2.3.2 Complexity Analysis of Linear Discriminant Analysis

LDA can get some computational benefits from the special structure of W_{LDA} as shown in the following equations.

$$\begin{aligned} & XW_{LDA}X^T\mathbf{a} = \lambda XX^T\mathbf{a} \\ \Rightarrow \quad & U\Sigma V^T W_{LDA} V\Sigma U^T\mathbf{a} = \lambda U\Sigma\Sigma U^T\mathbf{a} \\ \Rightarrow \quad & \Sigma^{-1}U^TU\Sigma V^T W_{LDA} V\Big(\Sigma U^T\mathbf{a}\Big) = \lambda\Sigma^{-1}U^TU\Sigma\Big(\Sigma U^T\mathbf{a}\Big) \\ \Rightarrow \quad & V^T W_{LDA} V\mathbf{b} = \lambda\mathbf{b} \end{aligned} \tag{2.42}$$

$V \in \mathbb{R}^{n\times d}$ is right singular matrix of X and d is the rank of X. The i-th row vector of V corresponds to the data point $\mathbf{x}_i$ and we denote it as $\mathbf{z}_i$, $V = [\mathbf{z}_1, \cdots, \mathbf{z}_m]^T$. Let $\mathbf{z}_i^{(k)}$

denote the row vector of V which corresponds to $\mathbf{x}_i^{(k)}$. Define $\boldsymbol{\nu}^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{z}_i^{(k)}$ and $H = [\sqrt{l_1}\boldsymbol{\nu}^{(1)}, \cdots, \sqrt{l_c}\boldsymbol{\nu}^{(c)}] \in \mathbb{R}^{d\times c}$. Inspired by Eqn. (2.15), we have

$$\begin{aligned} V^T W_{LDA} V =& \sum_{k=1}^{c}\frac{1}{n_k}\left(\sum_{i=1}^{n_k}\mathbf{z}_i^{(k)}\sum_{i=1}^{n_k}(\mathbf{z}_i^{(k)})^T\right) \\ =& \sum_{k=1}^{c} n_k\, \boldsymbol{\nu}^{(k)}(\boldsymbol{\nu}^{(k)})^T \\ =& HH^T \end{aligned} \tag{2.43}$$

The above algebraic steps show that the LDA projective functions can be obtained by the SVD decomposition of X and calculating the eigenvectors of HH^T.

It is easy to check that the left singular vectors of X (column vectors of U) are the eigenvectors of XX^T and the right singular vectors of X (column vectors of V) are the eigenvectors of X^TX [73]. Moreover, if U or V is given, then we can recover the other via the formula $XV = U\Sigma$ and $U^TX = \Sigma V^T$. In fact, the most efficient SVD decomposition algorithm (i.e. *cross-product*) applies this strategy [73]. Specifically, if $n \geq m$, we compute the eigenvectors of XX^T, which gives us U and can be used to recover V; If $n < m$, we compute the eigenvectors of X^TX, which gives us V and can be used to recover U. Since the matrix H is of size $r \times c$, where r is the rank of X and c is the number of classes. In most of the cases, r is close to $\min(m, n)$ which is far larger than c. Thus, comparing to directly calculate the eigenvectors of HH^T, compute the eigenvectors of H^TH then recover the eigenvectors of HH^T can achieve a significant saving.

When $n \geq m$, the calculation of XX^T requires $\frac{1}{2}nm^2$ flam; Computing the eigenvectors of XX^T requires $\frac{9}{2}m^3$ flam [73, 36]; Recovering V from U requires nm^2 flam by assuming X is of full rank; Computing the eigenvectors of HH^T requires $\frac{1}{2}mc^2+\frac{9}{2}c^3+mc^2$ flam; Finally, calculating **a**'s from **b**'s requiring m^2c. When $n < m$, we have the similar

analysis. We conclude that the time complexity of LDA measured by flam is

$$\frac{3}{2}mnt + \frac{9}{2}t^3 + \frac{3}{2}tc^2 + \frac{9}{2}c^3 + t^2c$$

where $t = \min(m, n)$. Considering $c \ll t$, the time complexity of LDA can be written as $\frac{3}{2}mnt + \frac{9}{2}t^3 + O(t^2)$.

Chapter 3

Spectral Regression for Efficient Subspace Learning

The graph embedding view of subspace learning provides us a powerful platform to develop various kinds of dimensionality reduction algorithms. However, the high computational cost restricts these algorithms to be applied to large scale high dimensional data sets. In this Chapter, we describe our approach which can overcome this difficulty.

3.1 Spectral Regression

In order to solve the eigen-problem

$$XWX^T\mathbf{a} = \lambda XDX^T\mathbf{a} \tag{3.1}$$

efficiently, we use the following theorem:

Theorem 1 *Let* $\boldsymbol{y}$ *be the eigenvector of eigen-problem*

$$W\boldsymbol{y} = \lambda D\boldsymbol{y} \tag{3.2}$$

with eigenvalue λ*. If* $X^T\boldsymbol{a} = \boldsymbol{y}$*, then* $\boldsymbol{a}$ *is the eigenvector of eigen-problem in Eqn. (3.1) with the same eigenvalue* λ*.*

Proof We have $W\mathbf{y} = \lambda D\mathbf{y}$. At the left side of Eqn. (3.1), replace $X^T\mathbf{a}$ by $\mathbf{y}$, we have

$$XWX^T\mathbf{a} = XW\mathbf{y} = X\lambda D\mathbf{y} = \lambda XD\mathbf{y} = \lambda XDX^T\mathbf{a}$$

Thus, **a** is the eigenvector of eigen-problem Eqn. (3.1) with the same eigenvalue λ.

Theorem (1) shows that instead of solving the eigen-problem in Eqn. (3.1), the linear projective functions can be obtained through two steps:

1. Solve the eigen-problem in Eqn. (3.2) to get **y**.

2. Find **a** which satisfies $X^T\mathbf{a} = \mathbf{y}$. In reality, such **a** might not exist. A possible way is to find **a** which can best fit the equation in the least squares sense:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} (\mathbf{a}^T\mathbf{x}_i - y_i)^2 \tag{3.3}$$

   where y_i is the i-th element of **y**.

The advantages of this two-step approach are as follows:

1. Both W and D are sparse matrices and the top eigenvectors of eigen-problem in Eqn. (3.2) can be efficiently calculated with Lanczos algorithms [73]. Moreover, we will show later how this eigen-problem is *trivial* and the eigenvectors **y** can be directly obtained with a supervised graph matrix W.

2. There exist many efficient iterative algorithms (*e.g.*, LSQR [66]) that can handle very large scale least square problems.

In the situation that the number of samples is smaller than the number of features, the minimization problem (3.3) is *ill posed*. We may have infinitely many solutions to the linear equations system $X^T\mathbf{a} = \mathbf{y}$ (the system is underdetermined). The most popular way to solve this problem is to impose a penalty on the norm of **a**:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T\mathbf{x}_i - y_i\right)^2 + \alpha\|\mathbf{a}\|^2 \right) \tag{3.4}$$

This is so called regularization and is well studied in statistics. The regularized least square is also called ridge regression [41]. The $\alpha \geq 0$ is a parameter to control the amounts of shrinkage. Now we can see the third advantage of the two-step approach:

3 Since the regression is used as a building block, the regularization techniques can be easily incorporated and produce more stable and meaningful solutions, especially when there exist a large number of features [41].

Our above two-step approach essentially performs regression after the spectral analysis of the graph, we called it *Spectral Regression* (SR).

3.2 Theoretical Analysis

The regularized least squares in Eqn. (3.4) can be rewritten in the matrix form as:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left((X^T\mathbf{a} - \mathbf{y})^T (X^T\mathbf{a} - \mathbf{y}) + \alpha \mathbf{a}^T\mathbf{a} \right). \tag{3.5}$$

Requiring the derivative of right side with respect to **a** vanish, we get

$$\begin{aligned} (XX^T + \alpha I)\mathbf{a} &= X\mathbf{y} \\ \Rightarrow \ \mathbf{a} &= (XX^T + \alpha I)^{-1} X\mathbf{y} \end{aligned} \tag{3.6}$$

When $\alpha > 0$, this regularized solution will not satisfy the linear equations system $X^T\mathbf{a} = \mathbf{y}$ and **a** will not be the eigenvector of eigen-problem in Eqn. (3.1). It is interesting and important to see when (3.6) gives the exact solutions of eigen-problem (3.1). Specifically, we have the following theorem:

Theorem 2 *Suppose **y** is the eigenvector of eigen-problem in Eqn. (3.2), if **y** is in the space spanned by row vectors of X, the corresponding projective function **a** calculated in Eqn. (3.6) will be the eigenvector of eigen-problem in Eqn. (3.1) as α deceases to zero.*

Proof Suppose $rank(X) = r$, the SVD decomposition of X is

$$X = U\Sigma V^T$$

The **y** is in the space spanned by row vectors of X, therefor, **y** is in the space spanned by column vectors of V. Thus, **y** can be represented as the linear combination of the column vectors of V. Moreover, the combination is unique because the column vectors of V are linear independent. Suppose the combination coefficients are $b_1, \cdots, b_r$. Let $\mathbf{b} = [b_1, \cdots, b_r]^T$, we have:

$$V\mathbf{b}=\mathbf{y} \;\Rightarrow\; V^TV\mathbf{b}=V^T\mathbf{y} \;\Rightarrow\; \mathbf{b}=V^T\mathbf{y} \;\Rightarrow VV^T\mathbf{y}=\mathbf{y} \tag{3.7}$$

To continue our proof, we need introduce the concept of pseudo inverse of a matrix [67], which we denote as $(\cdot)^+$. Specifically, pseudo inverse of the matrix X can be computed by the following two ways:

$$X^+ = V\Sigma^{-1}U^T$$

and

$$X^+ = \lim_{\alpha\to 0}(X^TX + \alpha I)^{-1}X^T$$

The above limit exists even if X^TX is singular and $(X^TX)^{-1}$ does not exist [67]. Thus, the regularized least squares solution in Eqn. (3.6)

$$\mathbf{a} = \Big(XX^T + \alpha I\Big)^{-1}X\mathbf{y} \overset{\alpha\to 0}{=} (X^T)^+\mathbf{y} = U\Sigma^{-1}V^T\mathbf{y}$$

Combine with the equation in Eqn. (3.7), we have

$$X^T\mathbf{a} = V\Sigma U^T\mathbf{a} = V\Sigma U^TU\Sigma^{-1}V^T\mathbf{y} = VV^T\mathbf{y} = \mathbf{y}$$

By Theorem (1), **a** is the eigenvector of eigen-problem in Eqn. (3.1).

When the the number of features is larger than the number of samples, the sample vectors are usually linearly independent, *i.e.*, $rank(X) = n$. In this case, we will have a stronger conclusion which is shown in the following Corollary.

Corollary 3 *If the sample vectors are linearly independent,* i.e., $rank(X) = n$*, all the projective functions calculated by Eqn. (3.6) are the eigenvectors of eigen-problem in Eqn. (3.1) as* α *deceases to zero. These solutions are identical to those of SVD+LGE in Eqn. (2.41).*

Proof The matrices W and D are of size $n \times n$ and there are n eigenvectors $\{\mathbf{y}_j\}_{j=1}^{n}$ of eigen-problem (3.2). Since $rank(X) = n$, all these n eigenvectors $\mathbf{y}_j$ are in the space spanned by row vectors of X. By Theorem (2), all n corresponding $\mathbf{a}_j$ of SR in Eqn (3.6) are eigenvectors of eigen-problem in Eqn. (3.1) as α decreases to zero. They are

$$\mathbf{a}_j^{SR} = U\Sigma^{-1}V^T\mathbf{y}_j.$$

Consider the eigen-problem in Eqn. (2.40), since the n eigenvectors $\mathbf{y}_j$ are also in the space spanned by column vectors of V, eigenvector $\mathbf{b}_j$ will be the solution of linear equations system $V\mathbf{b}_j = \mathbf{y}_j$. The column vectors of V are linearly independent, thus $\mathbf{b}_j$ is unique and

$$\mathbf{b}_j = V^T\mathbf{y}_j.$$

Thus, the projective functions of SVD+LGE

$$\mathbf{a}_j^{SVD+LGE} = U\Sigma^{-1}\mathbf{b}_j = U\Sigma^{-1}V^T\mathbf{y}_k = \mathbf{a}_j^{SR}$$

3.3 Eigenvectors of Supervised Graph Matrices

Now let us study the eigenvectors of eigen-problem in Eqn. (3.2). We consider the case that the graph weight matrix W is constructed with the label information, *i.e.*, searching the p nearest neighbors of $\mathbf{x}_i$ among the points share the same label with $\mathbf{x}_i$.

Without loss of generality, we assume that the data points in $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ are ordered according to their labels. It is easy to check that the matrix W in these three algorithms has a block-diagonal structure

$$W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \tag{3.8}$$

where c is the number of classes, $W^{(k)} \in \mathbb{R}^{n_k \times n_k}$ and n_k is the number of samples in k-th class. We also have the D as the diagonal matrix. Thus, the eigenvalues and eigenvectors of $W\mathbf{y} = \lambda D\mathbf{y}$ are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros) [36]:

$$W^{(k)}\mathbf{y}^{(k)} = \lambda D^{(k)}\mathbf{y}^{(k)}.$$

It is straightforward to show that the above eigen-problem has an eigenvector $\mathbf{e}^{(k)} \in \mathbb{R}^{n_k}$ associated with the largest eigenvalue 1, where $\mathbf{e}^{(k)} = [1, 1, \cdots, 1]^T$ [26]. Thus the top c eigenvectors of eigen-problem in Eqn. (3.2) are

$$\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} n_i}, \underbrace{1, \cdots, 1}_{n_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{c} n_i}]^T. \tag{3.9}$$

These eigenvectors correspond to the same largest eigenvalue 1. Since 1 is a repeated

eigenvalue, we could just pick any other c orthogonal vectors in the space spanned by $\{\mathbf{y}_k\}$ in Eqn. (3.9), and define them to be our c eigenvectors [36]. The vector of all ones is naturally in the spanned space. This vector is useless since the responses of all the data points are the same. In reality, we can pick the vector of all ones as our first eigenvector and use Gram-Schmidt process to get the remaining $c-1$ orthogonal eigenvectors. The vector of all ones can then be removed.

For the W in LDA, we can easily see that all the elements of $W^{(k)}$ are equal to $1/n_k$. Thus the rank of $W^{(k)}$ is 1 and there is only one non-zero eigenvalue which is exactly 1. We have exactly c eigenvectors (or $c-1$ useful eigenvectors after Gram-Schmidt process) with respect to non-zero eigenvalue for eigen-problem in Eqn. (3.2). For the W in LPP and NPE, we can get more eigenvectors since the rank of $W^{(k)}$ is usually larger than 1. For a c class problem, previous studies [3][45] show that $c-1$ projective functions are usually enough.

Our above analysis shows that when W is constructed by integrating label information, the top $c-1$ eigenvectors of eigen-problem in Eqn. (3.2) can be directly obtained. Moreover, although the graphs used in LDA, LPP and NPE are different, the top $c-1$ eigenvectors of their graph matrices are the same. Thus the projective functions calculated in SR are the same. By Theorem 2 and Corollary 7, these projective functions are identical to those of SVD+LGE approach in Eqn. (2.40) when the sample vectors are linearly independent. Our analysis here gives the reason why the three algorithms LDA [3], LPP [47] and NPE [45] achieve similar performance for high-dimensional low sample size problems.

It is easy to check that the values of the i-th and j-th entries of any vector $\mathbf{y}$ in the space spanned by $\{\mathbf{y}_k\}$ in Eqn. (3.9) are the same as long as $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class. Thus the i-th and j-th rows of Y are the same, where $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_{c-1}]$. Corollary (7) shows that when the sample vectors are linearly independent, the $c-1$ projective functions of LDA (LPP, NPE) are exactly the solutions of the $c-1$ linear equations systems $X^T\mathbf{a}_k = \mathbf{y}_k$. Let $A = [\mathbf{a}_1, \cdots, \mathbf{a}_{c-1}]$ be the transformation matrix which embeds the data points into

the LDA (LPP, NPE) subspace as:

$$A^T X = Y^T.$$

The columns of matrix Y^T are the embedding results of samples in the LDA (LPP, NPE) subspace. Thus, the data points with the same label are corresponding to the same point in the LDA (LPP, NPE) subspace when the sample vectors are linearly independent.

These projective functions are optimal in the sense of separating training samples with different labels. However, they usually overfit the training set thus may not be able to perform well for the test samples, thus the regularization is necessary.

3.4 Computational Complexity Analysis

SR uses regularized least squares to find the projective functions, which is a necessary step in both supervised and unsupervised cases. Thus, we begin our analysis with analyzing this step.

When m is not very large, the regularized least squares problem in Eqn. (3.4) can be solved by directly solving the linear equations system in Eqn. (3.6). The calculation of XX^T requires $\frac{1}{2}nm^2$ flam. Since the matrix $XX^T + \alpha I$ is positive definite, using Gaussian Elimination to solve the linear equations system in Eqn. (3.6) costs $\frac{1}{6}m^3$ flam [72].

For large scale high dimensional data, the regularized least squares problem in Eqn. (3.4) can be efficiently solved by iterative algorithm LSQR which is designed to solve large scale sparse linear equations and least squares problems [66]. In each iteration, LSQR needs to compute two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$. The remaining work load of LSQR in each iteration is $3n + 5m$ flam [65]. Thus, the time cost of LSQR in each iteration is $2mn + 3n + 5m$. If LSQR stops after k_2 iterations[1], the time cost is $k_2(2mn + 3n + 5m)$. Finally, the total time cost for d projective functions is $dk_2(2mn +$

[1]LSRQ converges very fast [66]. In our experiments, 20 iterations are enough.

Table 3.1: Computational complexity of LDA, LPP and SR

Time complexity (operation counts, *flam*)				
Algorithm		Graph Construction	Responses Generation	Embedding Functions
Supervised	LDA	–	–	$\frac{3}{2}mnt + \frac{9}{2}t^3$
	SR		nc^2	$2ck_2ns + 5ck_2m$
Unsupervised	LPP	$n^2(s + \log n)$	–	$\frac{3}{2}mnt + \frac{9}{2}t^3 + \min(\frac{9}{2}t, dk_1)t^2$
	SR		$dk_1n(p + 8)$	$2dk_2ns + 5dk_2m$

Memory cost		
Algorithm		
Supervised	LDA	$ns + (m + n)t + mc$
	SR	$ns + nc + mc$
Unsupervised	LPP	$ns + np + (m + n)t + md$
	SR	$ns + np + nd + md$

n: the number of data samples
m: the number of features
t: $\min(m, n)$
s: the average number of nonzero features for one sample ($s \leq n$)
c: the number of classes (LDA and SR will produce $c - 1$ projective functions)
d: the number of dimensions (projective functions) required in LPP and SR
p: the number of nearest neighbors
k_1: the number of iterations in Lanczos
k_2: the number of iterations in LSQR

$3n + 5m)$. Besides data matrix X, LSQR needs $n + 2m$ additional memory [65]. Finally, the memory cost in this step is $mn+n+2m+dm$, with dm to store the projective functions.

In supervised case, the eigen-problem in third step of SR is trivial and we can directly obtain those $c - 1$ eigenvectors. The cost of this step is mainly the cost of Gram-Schmidt method, which requires $(nc^2 - \frac{1}{3}c^3)$ flam and $nc + c^2$ memory [72].

In unsupervised case, the affinity graph construction step is same as we analyzed before. Since the p-nearest neighbor graph matrix W is sparse (has around np non-zero entries), we can use Lanczos algorithm to compute the first d eigenvectors within $dk_1n(p+8)$ flam, where k_1 is the iteration number for Lanczos algorithm. The memory requirement of this step is simply the memory to store W and d eigenvectors.

We summarize our complexity analysis results in Table 3.1. We assume $m \gg c$ and

only show the dominant part of the time and memory costs for simplicity. The main conclusions include:

- In supervised case:
  - ◇ LDA has cubic-time complexity with respect to $\min(m, n)$. Moreover, the left and right singular vector matrices of X, which are required to be stored in memory, are both dense. When both m and n are large, it is not feasible to apply LDA.
  - ◇ SR has linear-time complexity with respect to both m and n. It only has very small additional memory requirement besides data matrix X. Thus, SR can be easily scaled to high dimensional large data sets.
  - ◇ The computational complexity analysis clearly shows the advantages of using SR instead of directly applying LDA.
- In unsupervised case:
  - ◇ The graph construction step is unavoidable for all the spectral graph embedding approaches. If the same graph is used, the computational cost on this step can be neglected when we compare the different algorithms.
  - ◇ The popular manifold learning algorithm (*e.g.*, LLE, Isomap, Laplacian Eigenmaps) only compute the embedding results of the training data, which is exactly the responses generation step of SR. SR uses regression to find the projective functions with the additional linear-time complexity cost (with respect to both m and n) and almost no additional memory requirement.
  - ◇ Those linear (kernel) extension approaches (*e.g.*, LPP, NPE, Kernel Eigenmaps) directly calculate the projective functions by solving dense eigen-problems. They require additional cubic-time complexity cost (with respect to $\min(m, n)$)

and $(m+n) \cdot \min(m,n)$ memory cost. When both m and n are large, it is infeasible to apply these approaches.

- In both cases:
  - ◇ In many real problems, the data matrix is sparse. However, LDA and LPP need the **complete** SVD decomposition, which can not get any benefit from the sparseness of the data matrix. Moreover, the left and right singular matrices are both dense. They can not be fit into the memory when both m and n are large.
  - ◇ As shown in Table (4.1), SR can fully explore the sparseness of the data matrix and gain significant computational saving on both time and memory. SR can successfully applied as long as the data matrix X can be fit into the memory.
  - ◇ Even the data matrix X is too large to be fit into the memory, SR can still be applied with some reasonable disk I/O. This is because in each iteration of LSQR, we only need to calculate two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$, which can be easily implemented with X and X^T stored on the disk.

3.5 Experimental Results

In this section, we briefly show the experimental results of SR for supervised learning task (face recognition), unsupervised learning task (document clustering) and semi-supervised learning task (content-based image retrieval).

All of our experiments have been performed on an Intel Pentium D 3.20GHz Linux machine with 2GB memory. For the purpose of reproducibility, we provide our algorithms and data sets used in these experiments at:

http://www.cs.uiuc.edu/homes/dengcai2/Data/data.html

3.5.1 Face Recognition

In this section, we investigate the performance of our proposed SR approach for face recognition on PIE database.

The CMU PIE face database[2] contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. We choose the five near frontal poses (C05, C07, C09, C27, C29) and use all the images under different illuminations and expressions, thus we get 170 images for each individual. All the face images are manually aligned and cropped. The cropped images are 64×64 pixels, with 256 gray levels per pixel. The features (pixel values) are then scaled to [0,1] (divided by 256). For each individual, $l(= 30, 40, 50, 60, 80, 100, 120)$ images are randomly selected for training and the rest are used for testing.

The face recognition task is handled as a multi-class classification problem $-$ we map each test image to a low-dimensional subspace via the embedding learned from training data, and then classify the test data by the nearest neighbor classifier. Three subspace learning algorithms are compared in the experiment. They are:

1. Linear Discriminant Analysis (LDA)

2. Regularized Linear Discriminant Analysis (RDA)

3. Spectral Regression (SR)

The recognition error rates and the computational time are reported on the Table 3.2 and Figure 3.1. Considering both accuracy and efficiency, SR is the best choice among three of the compared approaches. It provides an efficient and effective regularized subspace learning solution for large scale data sets.

[2]http://www.ri.cmu.edu/projects/project_418.html

Table 3.2: Performance comparisons on PIE

	Error rates (mean±std-dev%)			Computational time (s)		
	LDA	RDA	SR	LDA	RDA	SR
G30/P140	8.8±0.3	5.9±0.3	6.1±0.2	59.37	396.2	17.39
G40/P130	8.6±0.2	5.0±0.2	5.2±0.2	131.2	404.5	20.11
G50/P120	9.3±0.4	4.6±0.3	4.8±0.3	241.3	413.1	22.71
G60/P110	10.1±1.2	4.2±0.2	4.5±0.2	394.9	421.8	25.49
G80/P90	7.5±0.2	3.9±0.2	4.2±0.2	442.1	442.1	31.13
G100/P70	6.2±0.2	3.7±0.2	4.0±0.2	455.4	455.4	35.98
G120/P50	5.6±0.3	3.5±0.2	3.8±0.2	471.6	471.6	41.57

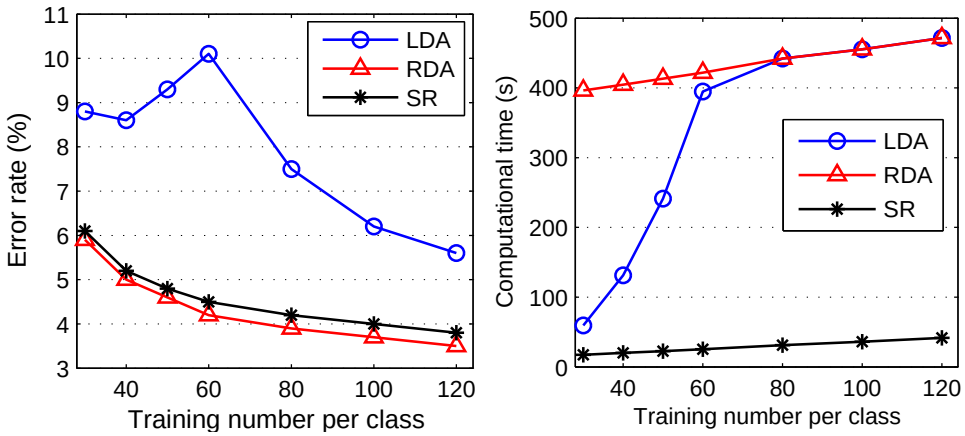


Figure 3.1: Recognition error rates and computational time of each algorithm on PIE.

3.5.2 Document Clustering

Clustering is one of most crucial techniques to organize the documents in an unsupervised manner. The ordinary clustering algorithms (*e.g.* K-means) can be performed in the original document space or in the reduced document space (by using the dimensionality reduction algorithms, *e.g.*, Latent Semantic Indexing (LSI)[29], LPP). In this experiment, we investigate the use of dimensionality reduction algorithms for text clustering. The following six methods are compared in the experiment:

- K-means on original term-document matrix, which is treated as our baseline (denoted as **Baseline**)
- K-means after Latent Semantic Indexing [29] (denoted as **LSI**)

- K-means after Locality Preserving Indexing [43] (denoted as **LPI**)
- K-means after Spectral Regression (denoted as **SR**)
- Clustering using Probabilistic Latent Semantic Indexing [48] (denoted as **PLSI**).
- Nonnegative Matrix Factorization-based clustering [83] (denoted as **NMF**).

It is important to note that the two methods LPI and SR need to construct a graph on the documents. In this experiment, we use the same graph for these two methods and the parameter p (number of nearest neighbors) was set to 7. The parameter α in SR was set to 0.1.

All these algorithms are tested on the TDT2 corpus. The TDT2 corpus[3] consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs (CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In this experiment, those documents appearing in two or more categories were removed, and only the largest 30 categories were kept, thus leaving us with 9,394 documents in total.

The clustering result is evaluated by comparing the obtained label of each document with that provided by the document corpus. The accuracy (AC) is used to measure the clustering performance [10], [83]. Given a document $\mathbf{x}_i$, let r_i and s_i be the obtained cluster label and the label provided by the corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum_{i=1}^{n} \delta(s_i, map(r_i))}{n}$$

where n is the total number of documents and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise, and map(r_i) is the permutation mapping function that maps each cluster label r_i to the equivalent label from the data corpus. The best mapping can be found by using the Kuhn-Munkres algorithm [56].

[3]Nist Topic Detection and Tracking corpus at http://www.nist.gov/speech/tests/tdt/tdt98/index.htm

Table 3.3: Clustering results on TDT2

c	Accuracy (mean±std-dev%)					
	Baseline	LSI	PLSI	LPP	SR	NMF
2	97.7±7.3	93.4±14.2	91.7±13.0	**99.8±0.3**	**99.9±0.2**	99.2±4.7
3	88.4±18.0	86.1±20.0	82.8±18.3	**99.6±0.4**	**99.6±0.4**	95.7±11.0
4	85.7±18.9	79.2±21.2	75.4±19.3	**99.3±0.8**	**99.4±0.8**	92.4±11.9
5	82.4±17.8	76.8±22.3	72.5±18.0	**98.7±1.8**	**98.8±1.8**	92.2±10.5
6	79.0±17.5	72.0±19.4	68.2±16.8	**98.6±1.5**	**98.8±1.3**	88.0±12.6
7	74.5±16.5	65.9±18.1	64.0±14.1	97.8±2.4	**98.2±2.1**	83.1±14.6
8	70.1±17.9	61.3±18.0	61.1±15.2	96.8±4.2	**97.3±4.2**	79.7±13.1
9	72.3±15.6	64.5±18.0	62.2±11.5	95.5±6.0	**97.5±2.4**	84.8±13.1
10	69.2±17.0	63.4±18.0	61.1±13.2	94.0±6.3	**96.0±4.5**	81.5±10.1
30	58.5	54.2	59.6	$-^*$	**86.7**	61.0

c	Processing time (s)					
	Baseline	LSI	PLSI	LPP	SR	NMF
2	6.25	0.43	3.22	11.50	1.08	6.0
3	18.23	0.67	7.49	26.81	1.95	23.1
4	29.74	0.96	11.23	33.56	2.62	63.3
5	61.82	1.50	18.67	76.37	4.50	113.7
6	66.51	1.78	20.70	65.30	4.37	238.4
7	117.63	2.97	31.57	143.86	7.65	389.5
8	171.76	4.20	40.06	179.03	9.27	766.6
9	193.85	5.00	45.57	228.12	10.93	869.7
10	261.05	6.48	56.79	266.53	13.09	1348.3
30	2720.21	132.12	511.53	$-^*$	224.71	15101.0

*LPI can not be applied due to the memory limit

Besides clustering the whole data set into 30 clusters, the evaluations were also conducted with different number of clusters, ranging from 2 to 10. For each given cluster number k, 50 tests were conducted on different randomly chosen categories, and the average performance was computed over these 50 tests (except the 30 cluster case). For each test, K-means algorithm was applied 10 times with different start points and the best result in terms of the objective function of K-means was recorded. After LSI, LPI, or SR, how to determine the dimensions of the subspace is still an open problem. In this experiment, we keep k dimensions for all the three algorithms as suggested by previous study [10].

Table 3.3 and Figure 3.2 show the average accuracy of the six algorithms. LSI seems not promising in dimension reduction for clustering because the K-means on the LSI subspace is even worse than K-means on the original document space. One may iterate all the

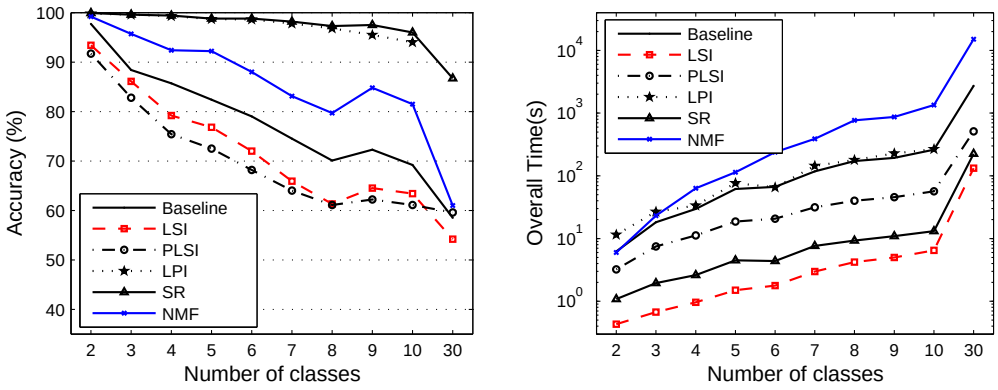


Figure 3.2: Clustering performance comparisons on TDT2 corpus

possible dimensions for better performance of LSI as suggested in [10]. However, it may not possible to do so in a real case. Clustering using PLSI is even worse. NMF method achieves better performance than Baseline which is consistent with previous study [83], [10]. Both LPI and SR achieve significant improvements over other four algorithms. The reason is that LPI and SR try to reveal the local geometric structure of document space. More detailed analysis and experiments of document clustering using LPI are provided in [10].

Table 3.3 and Figure 3.2 also show the processing time of the six algorithms. The processing time of LSI, LPI and SR include two parts: dimensionality reduction time and time of K-means on the reduced subspace. The processing time of Baseline and NMF methods are simply the time of clustering approaches (K-means and nonnegative matrix factorization). PLSI estimate the probability of each document belongs to each cluster, which can be directly used to infer the clustering result. Thus, the processing time of PLSI is only the dimensionality reduction (model estimation) time. After dimensionality reduction of LSI (LPI and SR), K-means is performed in a very low dimensional subspace thus is much more efficient then K-means in the original document space. The results here further show the advantage of dimensionality reduction for clustering. Clustering based on LSI is the most efficient approach. However, the low clustering accuracy makes LSI approach less attractable. Although the NMF method achieves better performance than

Table 3.4: Image features used in the experiment

Feature Name	Dimension
Color Histogram [63]	166
Color Correlogram [50]	144
Color Moment [74]	9
Wavelet Texture [1]	18
Canny Edge [22]	72
All	409

Baseline method, the high computational cost (NMF spent more than 4 hours for clustering 9,394 documents into 30 classes!) makes it not applicable on large document set. The same shortcoming exists for LPI approach. It can not be applied with 9,394 documents due to the memory limit. Consider both accuracy and efficiency, SR is obviously the best among the six compared algorithms for document clustering.

3.5.3 Content-Based Image Retrieval

In this section, we describe how to apply Spectral Regression to CBIR. Particularly, we consider relevance feedback driven image retrieval.

Features for Image Retrieval

Low-level image representation is a crucial problem in CBIR. General visual features includes color, texture, shape, etc. Color and texture features are the most extensively used visual features in CBIR. Compared with color and texture features, shape features are usually described after images have been segmented into regions or objects. Since robust and accurate image segmentation is difficult to achieve, the sue of shape features for image retrieval has been limited to special applications where objects or regions are readily available. In this work, we use a 409-dimensional features as shown in Table (3.4) which combines color, texture and shape infomration.

In fact, if the low-level visual features are accurate enough, that is, if the Euclidean

distances in the low-level feature space can accurately reflect the semantic relationship between images, then one can simply perform nearest neighbor search in the low-level feature space and the retrieval performance can be guaranteed. Unfortunately, there is no strong connection between low-level visual features and high-level semantic concepts based on the state-of-the-art computer vision techniques. Thus, one has to resort to user interactions to discover the semantic structure in the data.

Relevance Feedback Image Retrieval

Relevance feedback is one of the most important techniques to narrow down the gap between low level visual features and high level semantic concepts [69]. Traditionally, the user's relevance feedbacks are used to update the query vector or adjust the weighting of different dimensions. This process can be viewed as an on-line learning process in which the image retrieval system acts as a learner and the user acts as a teacher. The typical retrieval process is outlined as follows:

1. The user submits a query image example to the system. The system ranks the images in database according to some pre-defined distance metric and presents to the user the top ranked images.

2. The user provides his relevance feedbacks to the system by labeling images as "relevant" or "irrelevant".

3. The system uses the user's provided information to re-rank the images in database and returns to the user the top images. Go to step 2 until the user is satisfied.

All the subspace learning algorithms (*e.g.*, LPP and SR) can use the user's relevance feedbacks to update their graphs, which leads to better subspace for semantic concepts. Let $\mathbf{q}$ denote the query image and A be the transformation matrix of one subspace learning algorithm, i.e. $\mathbf{x}'_i = A^T\mathbf{x}_i$ and $\mathbf{q}' = A^T\mathbf{q}$. The distance between $\mathbf{x}'_i$ and $\mathbf{q}'$ can be computed

as follows:

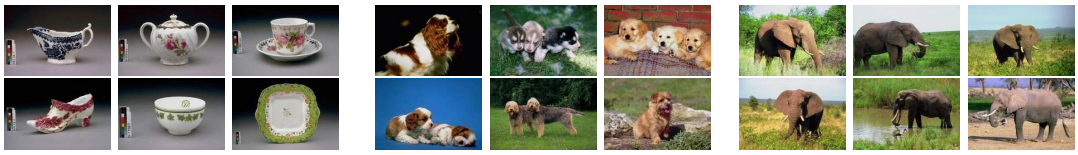
$$\begin{aligned} dist(\mathbf{x}'_i, \mathbf{q}') &= \sqrt{(\mathbf{x}'_i - \mathbf{q}')^T(\mathbf{x}'_i - \mathbf{q}')} \\ &= \sqrt{(\mathbf{x}_i - \mathbf{q})^T AA^T(\mathbf{x}_i - \mathbf{q})} \end{aligned}$$

For a general subspace learning algorithm, one needs to estimate the optimal dimensionality of the subspace which could be very hard in practical. Our analysis shows that there will be only c dimensions for SR subspace (with the Semi-supervised Discriminant Analysis formulation), where c is the number of classes. For image retrieval, $c = 2$ since there are two classes (relevant or not). Since all the other three suffer the problem of dimensionality estimation, this is one of the advantages of applying SR instead of other subspace learning algorithms.

In many situations, the number of images in the database can be extremely large, which makes the computation of all the algorithms infeasible. In order to reduce the computational complexity, we do not take all the images in the database to construct the p nearest neighbors graphs. Instead, we only take the top 400 images at the previous retrieval iteration, plus the labeled images, to find the optimal projection.

Image Data Set

The COREL data set is widely used in many CBIR systems, such as [42, 55, 77, 86]. For the sake of evaluations, we also choose this data set for testing. 30 categories of color images were selected, where each consists of 100 images. Each image is represented as a 409-dimensional vector as described before. Figure 3.3 shows some sample images from the COREL data set.



(a) (b) (c)

Figure 3.3: Sample images from category 24, 25, and 30, respectively.

Evaluation Metrics

Due to the relatively low recall in CBIR system, we do not use the *precision-recall* curve [51]. Instead, we use *precision-scope curve* and *precision rate* as the performance evaluation metrics [55]. The scope is specified by the number (N) of top-ranked images presented to the user. The precision is the ratio of the number of relevant images presented to the user to the scope N. The precision-scope curve describes the precision with various scopes and thus gives an overall performance evaluation of the algorithms. On the other hand, the precision rate emphasizes the precision at a particular value of scope.

In a real image retrieval system, a query image is usually not in the image database. To simulate such environment, we use *five-fold cross validation* to evaluate the algorithms which is also adopted in the paper [55]. More precisely, we divide the whole image database into five subsets with equal size. Thus, there are 20 images per category in each subset. At each run of cross validation, one subset is selected as the query set, and the other four subsets are used as the database for retrieval. The precision-scope curve and precision rate are computed by averaging the results from the five-fold cross validation.

Automatic Relevance Feedback Scheme

We designed an automatic feedback scheme to model the retrieval process. For each submitted query, our system retrieves and ranks the images in the database. The top 10 ranked images were selected as the feedback images, and their label information (relevant or irrelevant) is used for re-ranking. Note that, the images which have been selected at previous

iterations are excluded from later selections. For each query, the automatic relevance feedback mechanism is performed for four iterations. The similar scheme was used in [42], [55], [86].

Compared Algorithms

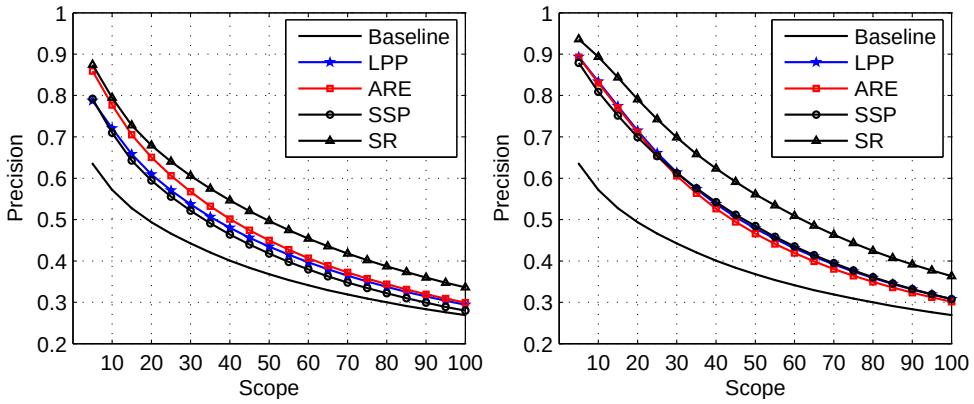
To demonstrate the effectiveness and efficiency of our proposed image retrieval algorithm (SR), we compare it with three state-of-the-art semi-supervised subspace learning algorithms, *i.e*. incremental Locality Preserving Projection (LPP) [42], Augmented Relation Embedding (ARE) [55] and Semantic Subspace Projection (SSP) [86].

A crucial problem of LPP (or, ARE and SSP) is how to determine the dimensionality of the subspace. In our experiments, we iterate all the dimensions and select the dimension with respect to the best performance. For SR, we simply use the 2-dimensional subspace. For all these algorithms, the Euclidean distances in the reduced subspace are used for ranking the images in the database. All these algorithms need to construct a k-nearest neighbors graph, we empirically set $k = 5$.

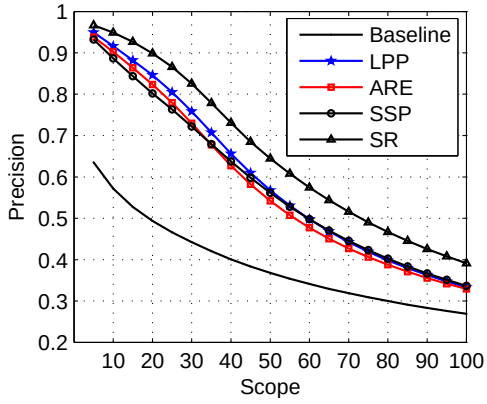
It is important to note that all the three algorithms (LPP, ARE and SSP) can be fit into the spectral regression framework to be efficiently computed. However, to show the advantages of SR, we implemented all the three algorithms in their ordinary ways (SVD+LGE approach).

Image Retrieval Performance

Figure 3.4 shows the average *precision-scope* curves of the different algorithms for the 1st, 2nd and 4th feedback iterations. The *baseline* curve describes the initial retrieval result without feedback information. Specifically, at the beginning of retrieval, the Euclidean distances in the original 409-dimensional space are used to rank the images in the database. After the user provides relevance feedbacks, the LPP, ARE, SSP, and SR algorithms are then applied to re-rank the images in the database. Our SR algorithm significantly outper-



(a) Feedback Iteration 1 (b) Feedback Iteration 2



(c) Feedback Iteration 4

Figure 3.4: Compare the retrieval performance of different algorithms. (a)-(c) Via illustrating with the precision-scope curves, we plot the results in the 1st, 2nd, and 4th feedback iteration, respectively. The SR algorithm performs the best on the entire scope for all the three feedback iterations.

forms the other three algorithms on the entire scope. The overall performances of LPP, ARE and SSP are very close to each other. ARE performs better than the other two at the first round, especially with a small scope. All these four algorithms are significantly better than the baseline, which indicates that the user provided relevance feedbacks are very helpful for improving the retrieval performance.

Table 3.5 gives the processing time for each query of the four algorithms. All the three algorithms LPP, ARE and SSP are computed by SVD+LGE approach as we described in Chapter 2. It is clear to see the SR has a significant computational advantage over the SVD+LGE approach. This results verified our theoretical analysis on computational

Table 3.5: Time on processing one query for each method (s)

	t_W	t_{SVD}	t_{GEigen}	t_{All}
LPP	0.062	0.453	0.494	1.009
ARE			0.489	1.004
SSP			0.487	1.002
		t_{SEigen}	t_{RLS}	
SR		0.024	0.041	0.127

t_W: time on the graph construction.
t_{SVD}: time on SVD decomposition.
t_{GEigen}: time on generalized eigen-problem.
t_{SEigen}: time on sparse eigen-problem
t_{RLS}: time on regularized least squares

complexity in Table 3.1.

Chapter 4

Kernel Spectral Regression

4.1 Derivation of LGE in Reproducing Kernel Hilbert Space

In this section, we generalize LGE approach to nonlinear problems and develop Kernel Graph Embedding (KGE).

We seek a function $f \in \mathcal{H}_K$ such that the following objective function is maximized,

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 W_{ij} \tag{4.1}$$

Proposition 4 *Let $\mathcal{H} = \{\sum_{i=1}^{n} \alpha_i K(\cdot, \mathbf{x}_i) | \alpha_i \in \mathbf{R}\}$ be a subspace of $\mathcal{H}_K$, the solution to the problem (4.1) is in $\mathcal{H}$.*

Proof Let $\mathcal{H}^{\perp}$ be the orthogonal complement of $\mathcal{H}$, i.e. $\mathcal{H}_K = \mathcal{H} \oplus \mathcal{H}^{\perp}$. Thus, for any function $f \in \mathcal{H}_K$, it has orthogonal decomposition as follows:

$$f = f_{\mathcal{H}} + f_{\mathcal{H}^{\perp}}$$

Now, let's evaluate f at $\mathbf{x}_i$:

$$\begin{aligned} f(\mathbf{x}_i) &= \langle f, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (4.2) \\ &= \langle f_{\mathcal{H}} + f_{\mathcal{H}_{\perp}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (4.3) \\ &= \langle f_{\mathcal{H}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} + \langle f_{\mathcal{H}^{\perp}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} && (4.4) \end{aligned}$$

Notice that $K_{\mathbf{x}_i} \in \mathcal{H}$ while $f_{\mathcal{H}^\perp} \in \mathcal{H}^\perp$. This implies that $\langle f_{\mathcal{H}^\perp}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} = 0$. Therefore,

$$f(\mathbf{x}_i) = \langle f_{\mathcal{H}}, K_{\mathbf{x}_i} \rangle_{\mathcal{H}_K} = f_{\mathcal{H}}(\mathbf{x}_i)$$

This completes the proof.

Since the solutions to the problem (4.1) are in $\mathcal{H}$, we use $\mathcal{H}_K$ and $\mathcal{H}$ interchangeably thereafter. Thus, the inner product between $f, g \in \mathcal{H}$ where $f(\cdot) = \sum_{i=1}^n \alpha_i K(\cdot, \mathbf{x}_i)$ and $g(\cdot) = \sum_{j=1}^n \beta_j K(\cdot, \mathbf{x}_j)$ is $\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$. If the kernel function is chosen as inner product $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$, then $\mathcal{H}_K$ is a linear functional space and the algorithm reduces to ordinary LGE. For general kernel function K and $f \in \mathcal{H}_K$, we have

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x})^T \alpha \tag{4.5}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^T$ and $K(\mathbf{x}) \doteq [K(\mathbf{x}, \mathbf{x}_1), \cdots, K(\mathbf{x}, \mathbf{x}_n)]^T$. We define

$$\begin{aligned}
\mathbf{y} &= (f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n))^T \\
&= \begin{pmatrix} K(\mathbf{x}_1)^T \boldsymbol{\alpha} \\ \vdots \\ K(\mathbf{x}_1)^T \boldsymbol{\alpha} \end{pmatrix} \\
&= \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \boldsymbol{\alpha} \\
&\doteq K\boldsymbol{\alpha}
\end{aligned}$$

where K is the kernel matrix, $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. Note that, K is symmetric. Thus, the

objective function can be reduced to

$$\begin{aligned}
& \sum_{i=1}^{n}(f(\mathbf{x}_i - f(\mathbf{x})_j))^2 W_{ij} \\
= \; & 2\sum_{i=1}^{n} f(\mathbf{x}_i) D_{ii} f(\mathbf{x}_i) - 2\sum_{i=1}^{n} f(\mathbf{x}_i) W_{ij} f(\mathbf{x}_j) \\
= \; & 2\mathbf{y}^T D\mathbf{y} - 2\mathbf{y}^T W\mathbf{y} \\
= \; & 2\mathbf{y}^T L\mathbf{y} \\
= \; & 2\boldsymbol{\alpha}^T KLK\boldsymbol{\alpha}
\end{aligned}$$

where L is the graph Laplacian. Similarly, the constraint can be derived as follows:

$$\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \boldsymbol{\alpha}^T KDK\boldsymbol{\alpha} = 1 \tag{4.6}$$

Therefore, the optimal mapping function $f \in \mathcal{H}_K$ can be obtained by solving the following minimization problem

$$\boldsymbol{\alpha}^* = \arg\min \frac{\boldsymbol{\alpha}^T KLK^T\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T KDK^T\boldsymbol{\alpha}}, \tag{4.7}$$

or equivalent maximization problem

$$\boldsymbol{\alpha}^* = \arg\max \frac{\boldsymbol{\alpha}^T KWK^T\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T KDK^T\boldsymbol{\alpha}}. \tag{4.8}$$

This leads to the following generalized eigenvector problem:

$$KWK\boldsymbol{\alpha} = \lambda KDK\boldsymbol{\alpha} \tag{4.9}$$

To get nonlinear function, we simply choose a nonlinear kernel. Also, it is important to note that there is no nonlinear optimization involved in KGE, hence it can be computed as simply as the standard LGE.

4.2 Efficient KGE via Spectral Regression

One can easily see the similarity between the eigen-problem of KGE in Eq. (4.9) and the eigen-problem of LGE in Eq. (2.34). Thus, the spectral regression idea introduced in Chapter 3 can also be applied on the KGE problem.

Specifically, we have the following theorm

Theorem 5 *Let* $\boldsymbol{y}$ *be the eigenvector of eigen-problem*

$$W\boldsymbol{y} = \lambda D\boldsymbol{y} \tag{4.10}$$

with eigenvalue λ*. If* $K\boldsymbol{\alpha} = \boldsymbol{y}$*, then* $\boldsymbol{\alpha}$ *is the eigenvector of eigen-problem in Eqn. (4.9) with the same eigenvalue* λ*.*

Proof We have $W\mathbf{y} = \lambda D\mathbf{y}$. At the left side of Eqn. (4.9), replace $K\boldsymbol{\alpha}$ by $\mathbf{y}$, we have

$$KWK\boldsymbol{\alpha} = KW\mathbf{y} = K\lambda D\mathbf{y} = \lambda KD\mathbf{y} = \lambda KDK\boldsymbol{\alpha}$$

Thus, $\boldsymbol{\alpha}$ is the eigenvector of eigen-problem Eqn. (4.9) with the same eigenvalue λ.

The above theorem shows that the KGE optimization problem can also be solved through regression. The kernel matrix K is positive semi-definite. When K is non-singular (positive definite), for any given $\mathbf{y}$, we have a unique $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$ which satisfy the above linear equations system. When K is singular, the system may have no solution or have infinite many solutions (the linear equations system is underdetermined) [36]. A possible way is to approximate $\boldsymbol{\alpha}$ by solving the following linear equations:

$$(K + \delta I)\boldsymbol{\alpha} = \mathbf{y} \tag{4.11}$$

where I is the identity matrix and $\delta \geq 0$ is the regularization parameter. Since the matrix $K + \delta I$ is positive definite, the Cholesky decomposition can be used to efficiently solve the

linear equations in Eqn. (4.11) [36], [72]. The computational complexity analysis will be provided in the later section.

The linear equations system in Eqn. (4.11) has close connection with regularized regression [79]. We denote the projective function in the feature space as:

$$f(\mathbf{x}) = \langle \boldsymbol{\nu}, \phi(\mathbf{x}) \rangle = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

It can be easily verified that the solution $\boldsymbol{\alpha}^* = (K + \delta I)^{-1}\mathbf{y}$ given by equations in Eqn. (3.3) is the optimal solution of the following regularized regression problem [79]:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \big(f(\mathbf{x}_i) - y_i\big)^2 + \delta \|f\|_K^2 \tag{4.12}$$

where y_i is the i-th element of $\mathbf{y}$, $\mathcal{F}$ is the RKHS associated with Mercer kernel $\mathcal{K}$ and $\| \ \|_K$ is the corresponding norm.

4.3 Theoretical Analysis

When the kernel matrix K is positive definite and the $\delta = 0$, Theorem 5 shows that the solution $\boldsymbol{\alpha}_k = K^{-1}\mathbf{y}$ are exactly the eigenvectors of the KGE eign-problem in Eqn. (4.9). In this case, Kernel Spectral Regression (KSR) is equivalent to ordinary KGE. Thus, it is interesting and important to see when the positive semi-definite kernel matrix K will be positive definite.

One of the most popular kernels is the Gaussian RBF kernel, $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$. Our discussion in this section will only focus on Gaussian kernel. Regarding the Gaussian kernel, we have the following lemma:

Lemma 6 (Full Rank of Gaussian RBF Gram Matrices [58]) *Suppose that* $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$

are distinct points, and $\sigma \neq 0$. The matrix K given by

$$K_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$$

has full rank.

Proof See [58] and Theorem 2.18 in [70].

In other words, the kernel matrix K is positive definite (provided no two $\mathbf{x}_i$ are the same).

Thus, we have the following theorem:

Theorem 7 *If all the sample vectors are different and the Gaussian RBF kernel is used, all the projective functions in KSR are eigenvectors of eigen-problem in Eqn. (4.9) when $\delta = 0$. In other words, the KSR and ordinary KGE are equivalent.*

Proof This theorem can be easily proofed by combining Lemma 6 and Theorem 5.

4.4 Computational Analysis

In this section, we provide the computational analysis of Kernel Spectral Regression. For simplicity, we use the LDA supervised graph in Eq. (2.16). In this case, KSR provides an efficient solution for Kernel Discriminant Analysis (KDA) [2][59]. We begin with the complexity analysis of the traditional KDA.

4.4.1 Computational Analysis of KDA

To get a stable solution of the eigen-problem in Eqn. (4.9), the matrix KDK is required to be non-singular [36]. When K is singular, there are two methods to solve this problem. The first method is by using eigen-decomposition of K, which was proposed in [2].

Suppose the rank of K is $r(r \leq n)$ and the eigen-decomposition of K is as follows:

$$K = U\Sigma U^T = U_r \Sigma_r U_r^T$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_n)$ is the diagonal matrix of sorted eigenvalues ($\sigma_1 \geq \cdots \geq \sigma_m \geq 0$) and U is the matrix of normalized eigenvectors associated to Σ. Σ_r is the diagonal matrix of nonzero eigenvalues and U_r is the first r columns of U. Thus Σ_r^{-1} exists and $U_r^T U_r = I$, where I is the identity matrix.

Substituting K in Eqn. (4.8)($D = I$ with W_{LDA}), we get

$$\boldsymbol{\alpha}_{opt} = \arg\max \frac{\left(\Sigma_r U_r^T \boldsymbol{\alpha}\right)^T U_r^T W U_r \left(\Sigma_r U_r^T \boldsymbol{\alpha}\right)}{\left(\Sigma_r U_r^T \boldsymbol{\alpha}\right)^T U_r^T U_r \left(\Sigma_r U_r^T \boldsymbol{\alpha}\right)}.$$

We proceed to variable modification using $\boldsymbol{\beta} = \Sigma_r U_r^T \boldsymbol{\alpha}$ and get:

$$\boldsymbol{\beta}_{opt} = \arg\max \frac{\boldsymbol{\beta}^T U_r^T W U_r \boldsymbol{\beta}}{\boldsymbol{\beta}^T \boldsymbol{\beta}},$$

Thus, the optimal $\boldsymbol{\beta}$'s are the leading eigenvectors of matrix $U_r^T W U_r$. Once $\boldsymbol{\beta}$'s are calculated, $\boldsymbol{\alpha}$ can be computed as $\boldsymbol{\alpha} = U_r \Sigma_r^{-1} \boldsymbol{\beta}$.

The second method is using the idea of regularization, by adding constant values to the diagonal elements of KK, as $KK + \gamma I$, for $\gamma > 0$. It is easy to see that $KK + \gamma I$ is nonsingular. This method is used in [59]. By noticing that

$$KK + \gamma I = U\Sigma U^T U \Sigma U^T + \gamma I = U(\Sigma^2 + \gamma I)U^T,$$

we define $\widetilde{\Sigma} = (\Sigma^2 + \gamma I)^{1/2}$, the objective function of regularized KDA can be written as:

$$\begin{aligned}
&\max \frac{\boldsymbol{\alpha}^T KWK\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T(KK+\gamma I)\boldsymbol{\alpha}} \\
=&\max \frac{\boldsymbol{\alpha}^T U\Sigma U^T WU\Sigma U^T\boldsymbol{\alpha}}{\boldsymbol{\alpha}^T U\widetilde{\Sigma}\widetilde{\Sigma}U^T\boldsymbol{\alpha}} \\
=&\max \frac{\boldsymbol{\beta}^T\widetilde{\Sigma}^{-1}\Sigma U^T WU\Sigma\widetilde{\Sigma}^{-1}\boldsymbol{\beta}}{\boldsymbol{\beta}^T\boldsymbol{\beta}}
\end{aligned}$$

where $\boldsymbol{\beta} = \widetilde{\Sigma}U^T\boldsymbol{\alpha}$. With this formulation, the above two methods can be computed in exactly the same way.

To reduce the computation in calculating $\boldsymbol{\beta}$, we shall exploit the special structure of W. Based on the analysis in the previous Section, we know that the matrix W has a block-diagonal structure. We partition the $n \times r$ matrix U_r as $[U_r^{(1)}, \cdots, U_r^{(c)}]^T$, where $U_r^{(k)} \in \mathbb{R}^{r\times n_k}$. Let $\mathbf{v}_i^{(k)}$ be the i-th column vector of $U_r^{(k)}$, we have:

$$\begin{aligned}
U_r^T WU_r =& \sum_{k=1}^{c} U_r^{(k)} W^{(k)} (U_r^{(k)})^T \\
=& \sum_{k=1}^{c} \frac{1}{n_k} \left(\sum_{i=1}^{n_k} \mathbf{v}_i^{(k)} \sum_{i=1}^{n_k} (\mathbf{v}_i^{(k)})^T \right) \\
=& \sum_{k=1}^{c} n_k \bar{\mathbf{v}}^{(k)} (\bar{\mathbf{v}}^{(k)})^T \\
=& HH^T
\end{aligned}$$

where $H = \left[\sqrt{n_1}\bar{\mathbf{v}}^{(1)}, \cdots, \sqrt{n_c}\bar{\mathbf{v}}^{(c)}\right] \in \mathbb{R}^{r\times c}$ and $\bar{\mathbf{v}}^{(k)}$ is the average vector of $\mathbf{v}_i^{(k)}$.

To calculate the c leading eigenvectors of HH^T, it is not necessary to work on matrix HH^T which is of size $r \times r$. We can use a much more efficient algorithm. Suppose the Singular Value Decomposition of H is

$$H = P\Gamma Q^T,$$

it is easy to check that the column vectors of P are the eigenvectors of HH^T and the column vectors of Q are the eigenvectors of H^TH [73]. Moreover, if P or Q is given, we can recover the other via the formula $HQ = P\Gamma$ and $P^TH = \Gamma Q^T$. Since $c \ll r$, we can calculate the c eigenvectors of H^TH and then recover the eigenvectors of HH^T, which are $\boldsymbol{\beta}$'s.

We use the term *flam* [72], a compound operation consisting of one addition and one multiplication, to measure the operation counts. All the kernel methods need to compute the kernel matrix K which requires $O(n^2m)$ flam, where m is the number of features. The eigen-decomposition of K requires $\frac{9}{2}n^3$ flam [73, 36]; Calculating the $c-1$ eigenvectors $\boldsymbol{\beta}$'s requires $\frac{9}{2}c^3 + \frac{3}{2}nc^2$ flam; Computing $\boldsymbol{\alpha}$'s from $\boldsymbol{\beta}$'s requires n^2c flam. Finally, we conclude the time complexity of KDA measured by flam is

$$\frac{9}{2}n^3 + n^2c + O(n^2m) + \frac{3}{2}nc^2 + \frac{9}{2}c^3.$$

Considering $n \gg c$, the above time complexity can be simplified as

$$\frac{9}{2}n^3 + n^2c + O(n^2m). \tag{4.13}$$

For a large scale problem, we have $n \gg m$. Thus, the time complexity of KDA is dominated by $\frac{9}{2}n^3$, which is the cost of eigen-decomposition of size $n \times n$ kernel matrix K.

4.4.2 Computational Analysis of KSR

The computation of KSR involves two steps: responses (**y** in Eqn. 3.9) generation and regularized regression. The cost of the first step is mainly the cost of Gram-Schmidt method, which requires $(nc^2 - \frac{1}{3}c^3)$ flam [72].

To solve the $c-1$ linear equations systems in Eqn. (3.3), we can use the Cholesky decomposition, which uniquely factorizes the positive definite matrix $K + \delta I$ in the form

$K+\delta I = R^T R$, where R is upper triangular with positive diagonal elements. The Cholesky decomposition requires $\frac{1}{6}n^3$ flam [72]. With this Cholesky decomposition, the $c-1$ linear equations can be solved within n^2c flam [72]. Besides solving the KSR optimization problem, we also need to compute the kernel matrix K which requires $O(n^2m)$ flam. Thus, the computational cost of KSR is

$$\frac{1}{6}n^3 + n^2c + O(n^2m) + nc^2 - \frac{1}{3}c^3,$$

which can be simplified as

$$\frac{1}{6}n^3 + n^2c + O(n^2m).$$

Comparing to the computational cost of ordinary KDA in Eqn. (4.13), KSR reduces the dominant part, which is $\frac{9}{2}n^3$ of ordinary KDA, to $\frac{1}{6}n^3$; achieves a 27-times speedup.

4.5 Incremental Kernel Discriminant Analysis

Due to the difficulty of designing an incremental solution for the eigen-decomposition on the kernel matrix in KDA, there has been little work on designing incremental KDA algorithms that can efficiently incorporate new data examples as they become available. The KSR algorithm uses regression instead of eigen-decomposition to solve the optimization problem, which provides us the chance to develop incremental version of KDA.

The major cost in KSR computation is the step of Cholesky decomposition which requires $\frac{1}{6}n^3$ flam. Fortunately, the Cholesky decomposition can be easily implemented in the incremental manner [72]. Actually, *Sherman's march*, one of the most popular Cholesky decomposition algorithms, is implemented in the incremental manner [72].

The procedure of Sherman's march is illustrated graphically in Figure 4.1. The gray area represents the part of the Cholesky decomposition that has already been computed with

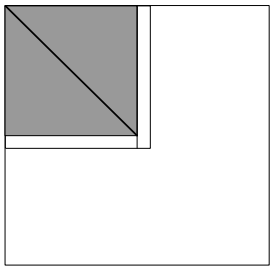


Figure 4.1: Sherman's march (Cholesky decomposition)

R and R^T separated by a diagonal line[1]. The white area represents untouched elements of the original matrix. The thin vertical box represents the column of R about to be computed. The algorithm is easy to derive. We show how to proceed from $(n-1)\times(n-1)$ submatrix to a $n \times n$ matrix. We have

$$
\begin{aligned}
K_n &= \begin{pmatrix} K_{n-1} & \mathbf{k}_{1n} \\ \mathbf{k}_{1n}^T & k_{nn} \end{pmatrix} \\
&= \begin{pmatrix} R_{n-1}^T & \mathbf{0} \\ \mathbf{r}_{1n}^T & r_{nn} \end{pmatrix} \begin{pmatrix} R_{n-1} & \mathbf{r}_{1n} \\ \mathbf{0} & r_{nn} \end{pmatrix},
\end{aligned}
$$

which leads to

$$
\begin{aligned}
K_{n-1} &= R_{n-1}^T R_{n-1} \\
\mathbf{k}_{1n} &= R_{n-1}^T \mathbf{r}_{1n} \\
k_{nn} &= \mathbf{r}_{1n}^T \mathbf{r}_{1n} + r_{nn}^2
\end{aligned}
$$

When the Cholesky decomposition of the $(n-1)\times(n-1)$ submatrix K_{n-1} is known, it is easy to get the Cholesky decomposition of the $n \times n$ K_n. For detailed derivation, please see [72].

Now, let us consider the additional computational cost of incremental KSR when Δn new data samples are injected to the system which already has n samples. Compare to the

[1] Actually, we only need to store R.

Table 4.1: Computational complexity of KDA and KSR

Algorithm		operation counts (*flam* [72])
Batch mode	KDA	$\frac{9}{2}n^3 + cn^2 + O(mn^2)$
	KSR	$\frac{1}{6}n^3 + cn^2 + O(mn^2)$
Incremental mode	KDA	$\frac{9}{2}n^3 + cn^2 + O(mn\Delta n)$
	KSR	$(\frac{\Delta n}{2} + c)n^2 + O(mn\Delta n)$

n: the number of data samples
m: the number of features
c: the number of classes
Δn: the number of new data samples

batch mode of KSR, we can get computational saving on two steps:

1. We only need to calculate the additional part of kernel matrix which requires $O(mn\Delta n + m\Delta n^2)$ flam;

2. The incremental Cholesky decomposition requires $\frac{1}{6}(n + \Delta n)^3 - \frac{1}{6}n^3$ flam [72].

Thus, the computation cost of incremental KSR measured by flam is

$$\frac{1}{2}n^2\Delta n + \frac{1}{2}n\Delta n^2 + \frac{1}{6}\Delta n^3 + (n + \Delta n)^2 c + O(mn\Delta n + m\Delta n^2) + (n + \Delta n)c^2 - \frac{1}{3}c^3.$$

When $\Delta n \ll n$ and $c \ll n$, the above cost can be simplified as

$$(\frac{\Delta n}{2} + c)m^2 + O(mn\Delta n).$$

We summarize our complexity analysis results in Table 4.1. The main conclusions include:

- The ordinary KDA needs to perform eigen-decomposition on the kernel matrix, which is very computationally expensive. Moreover, it is difficult to develop in-

Table 4.2: Statistics of the three data sets

dataset	dim (n)	train size (m)	test size	# of classes (c)
Isolet	617	6238	1559	26
USPS	256	7291	2007	10
PIE	1024	8000	3554	68

cremental algorithm based on the ordinary KDA formulation. In both batch and incremental modes, ordinary KDA has the dominant part of the cost as $\frac{9}{2}n^3$.

- KSR performs regression instead of eigen-decomposition. In the batch mode, it only has the dominant part of the cost as $\frac{1}{6}n^3$, which is a 27-times speedup of ordinary KDA. Moreover, it is easy to develop incremental version of KSR which only has quadratic-time complexity with respect to n. This computational advantage makes KSR much more practical in real world applications.

4.6 Experimental Results

In this section, we investigate the performance of our proposed KSR algorithm in both batch mode and incremental mode.

4.6.1 Datasets

Three datasets are used in our experimental study, including spoken letter, handwritten digit image, and face image data sets. The important statistics of three datasets are summarized below (see also Table 4.2):

- The Isolet spoken letter recognition database[2] was first used in [33]. It contains 150 subjects who spoke the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1 through isolet5.

[2]http://www.ics.uci.edu/∼mlearn/MLSummary.html

In the past usage [33][30], isolet1&2&3&4 were used as the training set and isolet5 was used as the test set. For the purposes of our experiment, we also choose isolet5 as the test set and perform several runs with isolet1, isolet1&2, isolet1&2&3, and isolet1&2&3&4 as the training set respectively.

- The USPS handwritten digit database is described in [52]. A popular subset [3] contains 9298 16×16 handwritten digit images in total, which is then split into 7291 training images and 2007 test images. In our experiment, we train all the algorithms on the first 1500 (3000, 4500, 6000, and 7291) images in the training set and test on the 2007 test images.

- The CMU PIE face database as we introduced in the previous Chapter.

4.6.2 Compared Algorithms

Four algorithms which are compared in our experiments are listed below:

1. Linear Discriminant Analysis (LDA) [35], which provides us a baseline performance of linear algorithms. We can examine the usefulness of kernel approaches by comparing the performance of KDA and LDA.

2. Kernel Discriminant Analysis (KDA) as discussed in Section 2. We test the regularized version and choose the regularization parameter δ by five fold cross-validation on the training set.

3. Kernel Spectral Regression (KSR), our approach proposed in this paper. The regularization parameter δ is also chosen by five fold cross-validation on the training set.

4. KDA/QR (KQR) [81], a KDA variation in which QR decomposition is applied rather than eigen-decomposition. Thus, KDA/QR is very efficient.

[3] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#usps

Table 4.3: Performance comparisons on Isolet dataset

	Error (%)					Time (s)					Speedup
Training Set	LDA	KDA	KSR	KQR	SVM	LDA	KDA	KSR	KQR	SVM	
Isolet1	15.27	11.74	12.89	17.19	12.51	1.93	18.86	1.21	0.93	4.75	15.6
Isolet1+2	6.61	3.79	3.85	7.63	4.11	2.14	134.6	5.51	3.60	13.79	24.4
Isolet1+2+3	5.90	2.99	3.08	7.12	3.34	2.37	451.6	14.09	7.98	23.84	32.1
Isolet1+2+3+4	5.71	2.82	2.89	6.86	3.27	2.56	991.2	27.86	14.02	34.82	35.6

*Column labeled "Speedup" shows how many times faster the KSR is (comparing to ordinary KDA).

Table 4.4: Performance comparisons on USPS dataset

	Error (%)					Time (s)					Speedup
Training Size	LDA	KDA	KSR	KQR	SVM	LDA	KDA	KSR	KQR	SVM	
1500	10.61	6.58	5.88	10.86	6.85	0.21	14.97	0.92	0.66	0.78	16.3
3000	9.77	5.53	5.38	10.66	5.58	0.27	111.9	4.35	2.61	2.20	25.7
4500	9.52	5.53	4.88	9.67	5.13	0.34	354.3	11.29	5.85	4.06	31.4
6000	9.92	5.03	4.43	9.37	5.08	0.40	825.3	22.74	10.41	6.22	36.3
7291	10.26	4.83	4.04	9.02	4.83	0.47	1553.6	37.59	15.60	8.18	41.3

5. Support Vector Machine (SVM) [79], which is believed as one of the state-of-the-art classification algorithms. Specifically, we use the LibSVM system [23] which implemented the multi-class classification with one versus one strategy. SVM is used to get the sense that how good the performance of KDA is.

We use the Gaussian RBF kernel for all the kernel-based methods. We tune the kernel width parameter σ and large margin parameter C in SVM to achieve best testing performance for SVM. Then, the same kernel width parameter σ is used in all the other kernel-based algorithms.

4.6.3 Results

The classification error rate as well as the training time (second) for each method on the three data sets are reported on the Table ($4.3 \sim 4.5$) respectively.

The main observations from the performance comparisons include:

- The Kernel Discriminant Analysis model is very effective in classification. KSR has

Table 4.5: Performance comparisons on PIE dataset

	Error (%)					Time (s)					Speedup
Training Size	LDA	KDA	KSR	KQR	SVM	LDA	KDA	KSR	KQR	SVM	
2000	5.29	5.18	4.81	15.62	6.30	8.77	36.51	2.47	1.66	24.13	14.8
3000	4.61	4.25	3.94	9.82	4.70	9.06	116.9	5.39	3.66	43.99	21.7
4000	4.14	5.53	3.24	7.93	3.74	9.42	256.6	10.35	6.39	68.43	24.8
5000	3.85	3.23	2.90	5.94	3.29	9.73	502.3	17.40	10.00	96.26	28.9
6000	3.57	2.91	2.53	5.68	2.84	10.06	830.7	27.21	14.20	125.6	30.5
7000	3.40	2.65	2.19	4.08	2.64	10.39	1340.9	38.65	19.12	155.6	34.7
8000	3.35	2.41	2.17	4.00	2.34	10.79	1908.1	53.75	24.96	186.7	35.5

the best performance for almost all the cases in all the three data sets (even better than SVM). For Isolet data set, previous study [30] reported the minimum error rate training on Isolet1+2+3+4 by OPT[4] with 30 bit ECOC is 3.27%. KDA (KSR) achieved better performance in our experiment for this train/test split. For USPS data set, previous studies [70] reported error rate 3.7% for KDA and 4.0% for SVM, slightly better than the results in our experiment. For all the cases, KDA (KSR) achieved significantly better performance than LDA, which suggests the effectiveness of kernel approaches.

- Since the eigen-decomposition of the kernel matrix is involved, the ordinary KDA is computationally expensive in training. KSR uses regression instead of eigen-decomposition to solve the optimization problem, and thus achieve significant speedup comparing to ordinary KDA. The empirical results are consistent with the theoretical estimation of the efficiency. The time of training KSR is comparable with that of training SVM. KSR is faster than SVM on Isolet and PIE data sets, while slower than SVM on USPS data set. This is because the time of training SVM is dependant with the number of support vectors [8]. For some data sets with lots of noise (*e.g.*, USPS), the number of support vectors is far less than the number of samples. In this case, SVM can be trained very fast.

[4]Conjugate-gradient implementation of back-propagation

- The KDA/QR algorithm is very efficient because it only need to perform QR decomposition on matrices with size $m \times c$ [81]. However, there is no theoretical relation between the optimization problem solved in KDA/QR and that of the KDA. In all the three data sets, the performances of KDA/QR is the worst.

4.6.4 Experiments on Incremental KDA

In this experiment, we study the computational cost of KSR performing in the incremental manner. The USPS and PIE data sets are used. We start from the training set with the size of 1000 (the first 1000 samples in whole training set) and increase the training size by 200 for each step. KSR is then performed in the incremental manner. It is important to note that KSR in the incremental manner give the exactly same projective functions as the KSR in the batch mode. Thus, we only care about the computational costs in this experiment.

Figure 4.2 and 4.3 shows log-log plots of how CPU-time of KDA (KSR, incremental KSR) increases with the size of the training set on USPS and PIE data set respectively. Lines in a log-log plot correspond to polynomial growth $O(n^d)$, where d corresponds to the slope of the line. The ordinary KDA scales roughly $O(n^{2.9})$, which is slightly better than the theoretical estimation. KSR in the batch mode has better scaling, which is also better than theoretical estimation with roughly $O(n^{2.6})$ over much of the range. This explains why KSR can be more than 27 times faster than ordinary KDA in the previous experiments. The KSR in the incremental mode has the best scaling, which is (to some surprise) better than quadratic with roughly $O(n^{1.8})$ over much of the range.

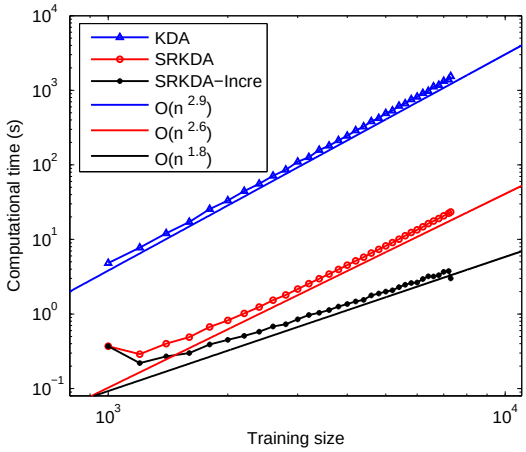


Figure 4.2: Computational cost of KDA, batch KSR and incremental KSR on the USPS data set.

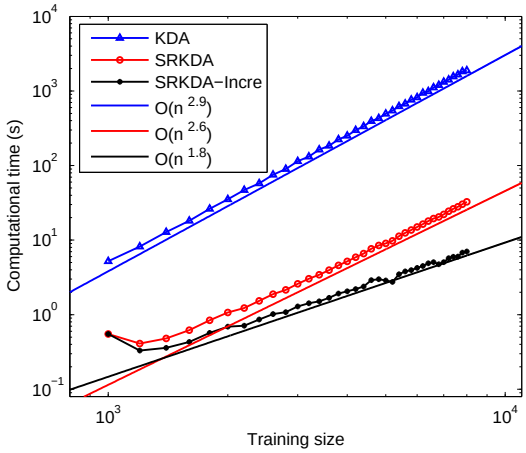


Figure 4.3: Computational cost of KDA, batch KSR and incremental KSR on the PIE data set.

Chapter 5

Sparse Subspace Learning for Feature Selection

One of the major disadvantages of all the algorithms discussed in previous Chapters is that the learned projective functions are linear combinations of all the original features, thus it is often difficult to interpret the results. Recently, there are considerable interests on developing sparse subspace learning algorithms. Zou *et al*. [89] proposed an elegant sparse PCA algorithm (SPCA) using their "Elastic Net" framework for L_1-penalized regression on regular principle components, solved very efficiently using *least angle regression* (LARS) [32]. Subsequently, d'Aspremont *et al*. [28] relaxed the hard cardinality constraint and solved for a convex approximation using semi-definite programming. In [60, 61], Moghaddam *et al*. proposed a spectral bounds framework for sparse subspace learning. Particularly, they proposed both exact and greedy algorithms for sparse PCA and sparse LDA.

In this Chapter, we propose a novel Unified Sparse Subspace Learning framework (USSL), for sparse projections learning. The proposed approach is fundamentally based on our spectral regression framework. By incorporating the regression as a building block, different kinds of regularizers can be naturally incorporated in SR. Specifically, with a L_1-norm regularizer (*lasso* or *elastic net*), the sparse projections can be efficiently computed in USSL.

5.1 Sparse Subspace Learning Formulation

For simplicity, we define $A = XWX^T$, $B = XDX^T$ and rewrite the optimization problem of LGE in Eqn. (2.9) as:

$$\begin{aligned} \textbf{max} \quad & \mathbf{a}^T A \mathbf{a} \\ \text{subject to} \quad & \mathbf{a}^T B \mathbf{a} = 1 \end{aligned}$$

Following [61], we define the Sparse Subspace Learning (SSL) optimization in terms of the following cardinality-constrained QCQP:

$$\begin{aligned} \textbf{max} \quad & \mathbf{a}^T A \mathbf{a} \\ \text{subject to} \quad & \mathbf{a}^T B \mathbf{a} = 1 \\ & \mathrm{card}(\mathbf{a}) = k \end{aligned} \tag{5.1}$$

The feasible set is all sparse $\mathbf{a} \in \mathbb{R}^m$ with k non-zero elements and $\mathrm{card}(\mathbf{a})$ as their L_0-norm. Unfortunately, this optimization problem is NP-hard and therefor generally intractable .

In [60, 61], Moghaddam *et al*. proposed a spectral bounds framework for sparse subspace learning. Particularly, they proposed both exact and greedy algorithms for sparse PCA and sparse LDA. Their spectral bounds framework is based on the following optimal condition of the sparse solution.

A sparse vector $\mathbf{a} \in \mathbb{R}^m$ with cardinality k yielding the maximum objective value in Eqn. (5.1) would necessarily imply that

$$\lambda_{max} = \frac{\mathbf{a}^T A \mathbf{a}}{\mathbf{a}^T B \mathbf{a}} = \frac{\mathbf{b}^T A_k \mathbf{b}}{\mathbf{b}^T B_k \mathbf{b}}$$

where $\mathbf{b} \in \mathbb{R}^k$ contains the k non-zero elements in $\mathbf{a}$ and the $k \times k$ principle sub-matrices of A and B obtained by deleting the rows and columns corresponding to the zero indices of $\mathbf{a}$. The k-dimensional quadratic form in $\mathbf{b}$ is equivalent to a standard unconstrained

generalized Rayleigh quotient, which can be solved by a generalized eigen-problem.

The above observation gives the exact algorithm for sparse subspace learning: a discrete search for the k indices which maximize λ_{max} of the subproblem (A_k, B_k). However, such observation does not suggest an efficient algorithm because an exhaustive search is still NP-hard. To solve this problem, Moghaddam *et al*. proposed an efficient greedy algorithm which combines *backward elimination* and *forward selection* [60, 61]. As we discussed in Section 2, many of the popular graph-based subspace learning algorithms can be formulated as the generalized eigen-problem, Moghaddam's approach provides a general solution for learning sparse projections in all these subspace learning algorithms. However, there are two major drawbacks of their approach:

1. Even their algorithm is a greedy one, the cost of backward elimination is with complexity $O(m^4 + nm^2)$[61].

2. In reality, more than one projective functions are usually necessary for subspace learning. However, the optimal condition of the sparse solution only gives the guide to find ONE sparse "eigenvector", which is the first projective function. It is unclear how to find the following projective functions. Although [60] suggests to use recursive deflation, the sparseness of the the following projective functions is not guaranteed.

In [89], Zou *et al*. proposed an elegant sparse PCA algorithm (SPCA) using their "Elastic Net" framework for L_1-penalized regression on regular principle components, solved very efficiently using *least angle regression* (LARS) [32]. The key idea of SPCA is formulating PCA as a regression-type optimization problem.

Without loss of generality, we assume the data are centered[1]. The PCA objective func-

[1]This can be achieved by subtracting the mean vector from all the sample vectors.

tion is

$$\begin{aligned} \textbf{max} \quad & \mathbf{a}^T X X^T \mathbf{a} \\ \text{subject to} \quad & \mathbf{a}^T \mathbf{a} = 1 \end{aligned} \tag{5.2}$$

and the optimal **a**'s are the eigenvectors with respect to the maximum eigenvalues of the following eigen-problem:

$$XX^T\mathbf{a} = \lambda\mathbf{a}. \tag{5.3}$$

Suppose the rank of X is r and the Singular Value Decomposition (SVD) of X is:

$$X = U\Sigma V^T, \tag{5.4}$$

it is easy to verify that the column vectors in U are the eigenvectors of XX^T [36], *i.e.*, the projective functions of PCA. Let $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_r] = U^T X = \Sigma V^T$, each row vector of Y is the sample vector in the r-dimensional PCA subspace. Thus, the projective functions of PCA are essentially the solutions of the linear equation systems:

$$X^T\mathbf{a}_t = \mathbf{y}_t, \quad t = 1, \cdots, r$$

in other words, $\mathbf{a}_t$ is the solution of the regression system:

$$\mathbf{a}_t = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} (\mathbf{a}^T\mathbf{x}_i - y_i^t)^2$$

where y_i^t is the i-th element of $\mathbf{y}_t$. Zou *et al.* [89] add L_1-regularizer to get the sparse solutions:

$$\mathbf{a}_t = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} (\mathbf{a}^T\mathbf{x}_i - y_i^t)^2 + \beta \sum_{j=1}^{m} |a_j|$$

where a_j is the j-th element of **a**. The above regression problem is called *Lasso* [41] and can be efficiently computed using LARS algorithm [32].

By using spectral regression framework, the similar technique can easily be applied to

those linear graph embedding algorithms.

5.2 Unified Sparse Subspace Learning via Spectral Regression

With a L_1-norm on **a** in the regression step of SR, we have

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{n} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \beta \sum_{j=1}^{m} |a_j| \right), \tag{5.5}$$

which is usually referred as *lasso regression* [41]. Due to the nature of the L_1 penalty, some coefficients will be shrunk to exact zero if β is large enough. Therefore the lasso produces a sparse model, which is exactly what we want. However, the lasso has several limitations as pointed out in [88]. The most relevant one to this work is that the number of selected features by the lasso is limited by the number of samples. For example, if applied to the face image data where there are thousands of features ($m > 1000$) with less than 100 samples ($n < 100$), the lasso can only select at most n features, which is clearly unsatisfactory. The Elastic Net [88] generalizes the lasso to overcome its drawbacks by combining both the ridge and lasso penalty:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{n} \left(\mathbf{a}^T \mathbf{x}_i - y_i \right)^2 + \alpha \sum_{j=1}^{m} a_j^2 + \beta \sum_{j=1}^{m} |a_j| \right) \tag{5.6}$$

For kernel subspace learning algorithms, recall the second step of KSR, which is solving the linear equations system $K\boldsymbol{\alpha} = \mathbf{y}$. Essentially, we try to solve a regression problem:

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \left(K(:, \mathbf{x}_i)^T \boldsymbol{\alpha} - y_i \right)^2$$

where $K(:, \mathbf{x}_i)$ is the i-th column of K and y_i is the i-th element of **y**. We can also use a

L_1-norm regularizer:

$$\min_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{n} \left(K(:, \mathbf{x}_i)^T \boldsymbol{\alpha} - y_i \right)^2 + \delta \sum_{i=1}^{n} |\alpha_i| \right) \tag{5.7}$$

In this way, Spectral Regression framework provides a natural sparse subspace learning approach.

5.3 Computational Complexity of USSL

The USSL computation involves two steps: responses generation (calculate the eigenvectors of eigen-problem in Eqn. (3.2)) and regularized regression.

For the W in LDA, the cost of the first step is mainly the cost of Gram-Schmidt method, which is $O(nc^2)$ [72]. For a k-NN graph W in LPP, the cost of the first step is $O(n^2m + n^2 \log n + qdnp)$. $O(n^2m)$ is used to calculate the pairwise distance between n samples with m features and $O(n^2 \log n)$ is used for p-nearest neighbors finding for all the n samples. The p-NN graph matrix W is sparse and the Lanczos algorithm [36] can be used to efficiently compute the first d eigenvectors of the eigen-problem in Eqn. (3.2) within $O(qdnp)$, where q is number of iterations in Lanczos.

All of the three types of regularized regression problems can be solved in $O(m^3+nm^2)$ [41][32]. By using the *Least Angel Regression* (LARS) algorithm [32], the entire solution path (the solutions with all the possible cardinality on **a**) of lasso and elastic net with a specific α can be computed in $O(m^3 + nm^2)$.

Considering $n \gg c$ and $n \gg d$, USSL provides a sparse LDA solution with $O(m^3 + nm^2)$ complexity and a sparse LPP solution with $O(n^2m + n^2 \log n + m^3 + nm^2)$ complexity. This complexity is exactly the same as the ordinary non-sparse solution solved by generalized eigen-problem. Comparing to the $O(m^4 + nm^2)$ greedy algorithm described in [61], USSL is much more efficient.

5.4 Experimental Results

In this section, we investigate the performance of our proposed USSL approach for both supervised learning (face recognition) and unsupervised learning (face clustering).

Two face databases were used in the experiment. The first one is the PIE (Pose, Illumination, and Experience) database[2] from CMU, and the second one is the Extended Yale-B database[3].

The CMU PIE face database contains 68 human subjects with 41,368 face images as a whole. The face images were captured by 13 synchronized cameras and 21 flashes, under varying pose, illumination and expression. We choose the frontal poses (C27) and use all the images under different illuminations and expressions, thus we get 3329 face images in total.

The Extended Yale-B face database contains 16128 images of 38 human subjects under 9 poses and 64 illumination conditions. In this experiment, we choose the frontal pose and use all the images under different illumination. Finally we get 2414 images in total.

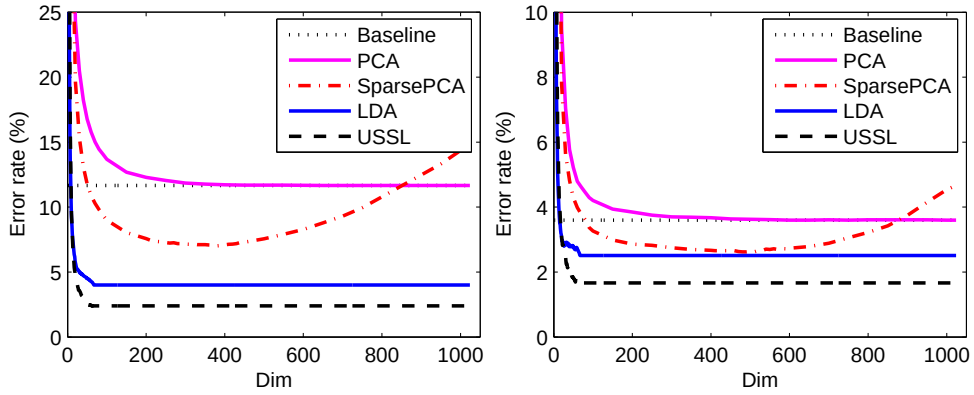
All the face images are manually aligned and cropped. The size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. Thus each image is represented as a 1024-dimensional vector.

5.4.1 USSL for Supervised Learning

In this experiment, we use the W in Eqn. (2.16). Thus, USSL provides a sparse LDA solution. We compare our algorithm with PCA, LDA and SparsePCA [89]. In face recognition, PCA and LDA are also called Eigenface [78] and Fisherface [3]. They are two of the most popular linear methods for face recognition. We do not compare with Sparse LDA [61] since it can only be applied to two-class case. Please refer to [61] for the details.

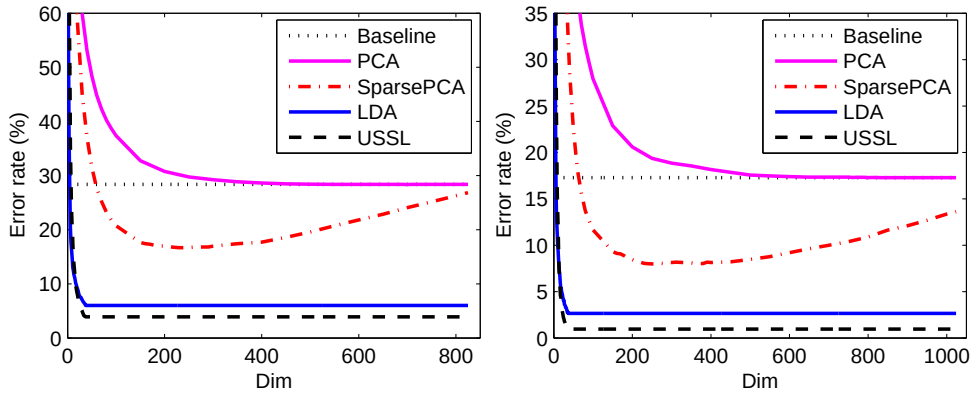
For each database, r $(= 33, 67)$ percent of samples are randomly selected for training

[2] http://www.ri.cmu.edu/projects/project_418.html
[3] http://vision.ucsd.edu/∼leekc/ExtYaleDatabase/ExtYaleB.html



(a) 33% Training (b) 67% Training

Figure 5.1: Error rate vs. dimensionality reduction on PIE database



(a) 33% Training (b) 67% Training

Figure 5.2: Error rate vs. dimensionality reduction on Yale-B database

and the rest are used for testing. The training samples are used to learn the basis functions. By using these basis functions, the testing images can be mapped into lower dimensional subspace where recognition is carried out by using nearest neighbor classifier. 5-fold cross validation has been performed in SparsePCA and USSL for selecting the best cardinality of the basis functions. The choices of the cardinality are 10, 20, $\cdots$ 100, 150, 200, $\cdots$, 1000, 1024.

For each given r, we average the recognition results over 20 random splits. Figure 5.1 and 5.2 show the plots of error rate versus dimensionality reduction for the PCA, SparsePCA, LDA, USSL and baseline methods on PIE and Yale-B databases, respec-

Table 5.1: Comparison of classification error rate on PIE

Method	33% Training			67% Training		
	error (%)	dim	sparsity	error (%)	dim	sparsity
Baseline	11.7±0.5	1024	–	3.6±0.6	1024	–
PCA	11.7±0.5	700	0	3.6±0.6	1000	0
SparsePCA	7.0±0.6	380	92.2%	2.6±0.5	480	92.2%
LDA	4.0±0.2	67	0	2.5±0.5	67	0
USSL	2.4±0.2	64	90.2%	1.6±0.3	66	90.2%

Table 5.2: Comparison of classification error rate on Yale-B

Method	33% Training			67% Training		
	error (%)	dim	sparsity	error (%)	dim	sparsity
Baseline	28.4±1.3	1024	–	17.3±0.7	1024	–
PCA	28.4±1.3	700	0	17.3±0.7	830	0
SparsePCA	16.7±1.1	230	95.1%	8.0±0.5	250	95.1%
LDA	6.0±0.6	37	0	2.7±0.5	37	0
USSL	3.9±0.6	37	86.3%	1.0±0.3	37	86.3%

tively. For the baseline method, the recognition is simply performed in the original 1024-dimensional image space without any dimensionality reduction. Note that, the upper bound of the dimensionality of LDA is $c - 1$ where c is the number of individuals [31]. We use the LDA graph W as defined in Section 2 in our USSL algorithm. Thus, the upper bound of the dimensionality of USSL is also $c - 1$. As can be seen, the performance of the PCA, SparsePCA, LDA and USSL algorithms varies with the number of dimensions. We show the best results together with the standard deviations obtained by them in Table 5.1 and 5.2 and the corresponding face subspaces are called optimal face subspace for each method. Particularly, we also shown the sparsity of the basis functions for these algorithms. The sparsity is computed as the ratio of the number of zero entries and the total number of entries. As can be seen, the sparsity for PCA and LDA are both zero, while the sparsity for sparse PCA and USSL are very high.

5.4.2 USSL for Unsupervised Learning

In this subsection, we investigate the use of our proposed approach for face clustering. Face clustering is an unsupervised task and we compare our algorithm with PCA, SparsePCA and Locality Preserving Projection (LPP) [46][47]. We use the same p-nearest neighbor graph in LPP and USSL. Thus, USSL provides a sparse LPP solution. We empirically set the value of p to 5.

We choose K-means as our clustering algorithm. K-means can be performed in the original feature space (Baseline) or in the reduced feature space (by using the dimensionality reduction algorithms, *e.g.*, PCA, LPP and USSL). The clustering result is evaluated by comparing the obtained label of each image with that provided by the ground truth. We use the normalized mutual information ($\overline{MI}$) to measure the clustering performance [10]. Let C denote the set of clusters obtained from the ground truth and C' obtained from an algorithm. Their mutual information metric $MI(C, C')$ is defined as follows:

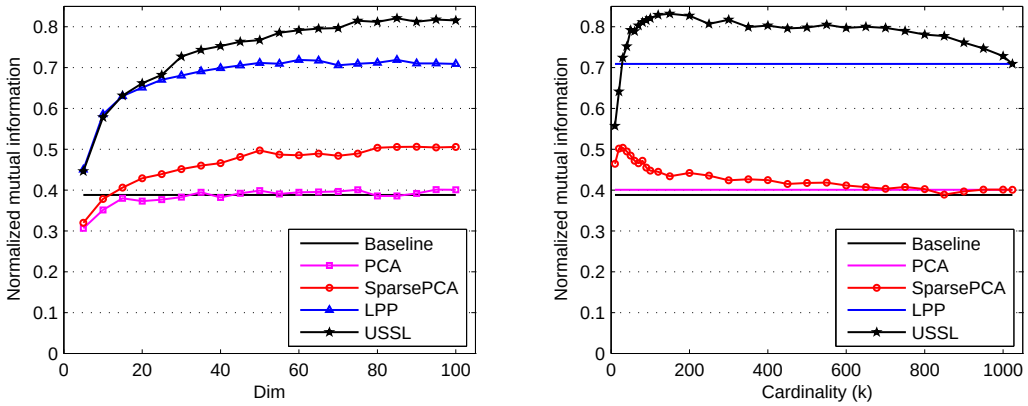
$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a sample arbitrarily selected from the data set belongs to the clusters c_i and c'_j, respectively, and $p(c_i, c'_j)$ is the joint probability that the arbitrarily selected document belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information $\overline{MI}$ as follows:

$$\overline{MI}(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))}$$

where $H(C)$ and $H(C')$ are the entropies of C and C', respectively. It is easy to check that $\overline{MI}(C, C')$ ranges from 0 to 1. $\overline{MI} = 1$ if the two sets of clusters are identical, and $\overline{MI} = 0$ if the two sets are independent.

Figure (5.3(a)) shows the plot of normalized mutual information versus dimensional-



(a) (b)

Figure 5.3: Normalized mutual information vs. dimensionality (a) and Normalized mutual information vs. cardinality (b) on PIE database

ity for the PCA, SparsePCA, LPP, USSL and baseline methods. As can be seen, all the methods obtain the best performance with dimensionality less than 100, and there is no performance improvement with more dimensions. Our USSL algorithm outperforms the other four methods. LPP performs the second best. PCA performs the worst, close to the baseline.

Figure (5.3(b)) shows the performances of all the algorithm in the 100-dimensional subspace. We show the performance change with the cardinality of basis functions in SparsePCA and USSL. As can be seen, the best performance is obtained with relatively small cardinality.

5.4.3 Experiments on Sparse KSR

In this experiment, we study the performance of KSR performing in the sparse mode, *i.e.*, the KSR with L_1-norm regularizer to produce the sparse KDA solution. To the best of our knowledge, there is no other published method to generate a sparse KDA solution. Moghaddam's sparse LDA approach [61] can be modified to generate the sparse KDA solution. However, as we pointed out before, their approach can only generate ONE sparse

Table 5.3: Classification error on Isolet dataset

	Error (%)			Sparsity
Training Set	KDA	KSR	KSR(Sparse)	
Isolet1	11.74	12.89	11.74	60%
Isolet1+2	3.79	3.85	3.59	60%
Isolet1+2+3	2.99	3.08	2.82	60%
Isolet1+2+3+4	2.82	2.89	2.82	60%

Table 5.4: Classification error on USPS dataset

	Error (%)			Sparsity
Training Set	KDA	KSR	KSR(Sparse)	
1500	6.58	5.88	5.83	60%
3000	5.53	5.38	5.13	60%
4500	5.53	4.88	4.73	60%
6000	5.03	4.43	4.04	60%
7291	4.83	4.04	3.94	60%

Table 5.5: Classification error on PIE dataset

	Error (%)			Sparsity
Training Set	KDA	KSR	KSR(Sparse)	
2000	5.18	4.81	4.73	60%
3000	4.25	3.94	3.71	60%
4000	5.53	3.24	3.12	60%
5000	3.23	2.90	2.81	60%
6000	2.91	2.53	2.44	60%
7000	2.65	2.19	2.17	60%
8000	2.41	2.17	2.14	60%

projective function and only suitable for binary class problem. While all the three data sets studied in this paper are multi-class data sets.

Three data sets used in this experiment are Isolet, USPS and PIE. Please see Chapter 4 for detailed description. Table (5.3), (5.4) and (5.5) show the classification error rate of KSR in sparse mode on the three data sets respectively. By using the *Least Angel Regression* (LARS) algorithm [32], the entire solution path (the solutions with all the possible cardinality on the projective function $\boldsymbol{\alpha}$) can be computed. After this, we use cross validation to select the optimal cardinality of the projective function in the experiment. We also show the sparsity of the projective function of KSR(sparse) in the tables. The sparsity

is defined as the percentage of zero entries in a projective vector. For ordinary KDA and KSR, the projective functions (vectors) are dense and the sparsity is zero.

As can be seen, the KSR(sparse) generates much more parsimonious model. The sparsity of the projective function in KSR(sparse)is 60%, which means the number of the "support vectors" are less than half of the total training samples. Moreover, such parsimony leads to better performance. In all the cases, the performance of KSR(sparse) is better than that of the ordinary KDA and KSR.

Chapter 6

Learning a Spatially Smooth Subspace for Face Recognition

We have discussed how to use both L_2 and L_1 norm regularizers in SR in previous several chapters. Both these two regularizers are data independent. In some real applications, one might hope that the characteristic of the data can help us to design specific regularizer. In this chapter, we will discuss how to design a specially smooth regularizer for face recognition.

The subspace learning algorithms have been extensively applied on face recognition [78][3][47]. All these methods consider a face image as a high dimensional vector. They do not take advantage of the spatial correlation of pixels in the image, and the pixels are considered as independent pieces of information. However, a $m_1 \times m_2$ face image represented in the plane is intrinsically a matrix, or 2-order tensor. Even though we have $m_1 \times m_2$ pixels per image, this spatial correlation suggests the real number of freedom is far less. Recently there have been a lot of interest in tensor based approaches to data analysis in high dimensional spaces. Vasilescu and Terzopoulos have proposed a novel face representation algorithm called Tensorface [80]. Tensorface represents the set of face images by a higher-order tensor and extends Singular Value Decomposition (SVD) to higher-order tensor data. Some other researchers have also shown how to extend PCA, LDA, LPP, MFA and LDE to higher order tensor data [11, 25, 44, 84, 85]. Some experimental results have showed the superiority of these tensor approaches over their corresponding vector approaches. However, our analysis later will show that these tensor approaches only consider the relationship between pixels in the same row (column) and fail to fully explorer the spatial information of images. The embedding functions of tensor approaches will still be spatially rough.

In this Chapter, we introduce a Spatially Smooth Subspace Learning (SSSL) model using a Laplacian penalty to constrain the coefficients to be spatially smooth. Instead of considering the basis function as a $m_1 \times m_2$-dimensional vector, we consider it as a matrix, or a discrete function defined on a $m_1 \times m_2$ lattice. Thus, the discretized Laplacian can be applied to the basis functions to measure their smoothness along horizontal and vertical directions. The discretized Laplacian operator is a finite difference approximation to the second derivative operator, summed over all directions. The choice of Laplacian penalty allows us to incorporate the prior information that neighboring pixels are correlated. Once we obtain compact representations of the images, classification and clustering can be performed in the lower dimensional subspace.

6.1 Graph Based Tensor Subspace Analysis

A face image represented in the plane is intrinsically a matrix, or the second order tensor. The relationship between nearby pixels of the image might be important for finding a projection. Recently there have been a lot of interest in extending the ordinary vector-based subspace learning approaches to tensor space [11, 25, 44, 84, 85].

The tensor-based approaches directly operate on the matrix representation of image data and are believed can capture the spatial relationship between the pixels. To examine what kind of spatial relationship has been captured in these tensor-based approaches, we need to examine the basis function.

Let $\{\mathbf{u}_k\}_{k=1}^{m_1}$ be an orthonormal basis of $\mathcal{R}^{m_1}$ and $\{\mathbf{v}_l\}_{l=1}^{m_2}$ be an orthonormal basis of $\mathcal{R}^{m_2}$. It can be shown that $\{\mathbf{u}_i \otimes \mathbf{v}_j\}$ forms a basis of the tensor space $\mathcal{R}^{m_1} \otimes \mathcal{R}^{m_2}$ [54]. Specifically, the projection of $T \in \mathcal{R}^{m_1} \otimes \mathcal{R}^{m_2}$ on the basis $\mathbf{u}_i \otimes \mathbf{v}_j$ can be computed as their inner product:

$$< T, \mathbf{u}_i \otimes \mathbf{v}_j >=< T, \mathbf{u}_i \mathbf{v}_j^T >= \mathbf{u}_i^T T \mathbf{v}_j$$

The ordinary vector-based approaches are linear, *i.e.*, $y_i = \mathbf{a}^T \mathbf{x}_i$ where $\mathbf{x}_i \in \mathbb{R}^m$ is the

vector representation of the i-th image, **a** is the projection vector (basis vector) and y_i is the one-dimensional embedding on this basis. The m values in basis function **a** are independently estimated. The tensor-based approaches are multilinear, *i.e.*, $y_i = \mathbf{u}^T T_i \mathbf{v}$, where $T_i \in \mathcal{R}^{m_1} \otimes \mathcal{R}^{m_2}$ is the *matrix* representation of the i-th image and $m = m_1 \times m_2$. The m values in a tensor basis $\mathbf{u}\mathbf{v}^T$ only have $m_1 + m_2$ degrees of freedom. In fact, the tensor-based approaches can be thought of as special cases of vector-based approaches with the following constraint:

$$a_{i+m_1(j-1)} = u_i v_j \tag{6.1}$$

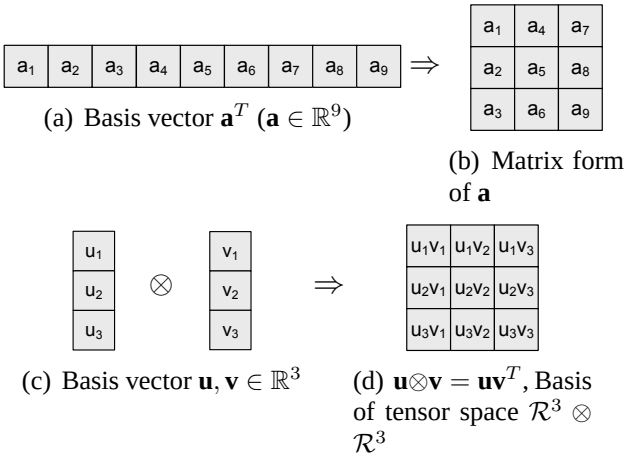
where a_i, u_i and v_i are the i-th elements in **a**, **u** and **v** respectively.

Figure (6.1) gives a intuitive example. It is easy to see that there is a common divisor of the values belong to the same row (or column) in a tensor basis, which exactly the spatial relation captured by the tensor-based approaches. Intuitively, the spatial correlation of pixels in a face image would suggest the spatial smoothness of the basis function, *i.e.*, the element values in basis function would be similar if the elements are spatially near. However, the tensor-based approaches have no guarantee on this and the basis function could still be spatially rough.

A more natural measurement of spatial smoothness of basis function could be the sum of the squared differences between nearby elements. In the next section, we will show how to achieve this by incorporating a 2-D discretized laplacian smoothing term in ordinary vector-based approaches.

6.2 Spatially Smooth Subspace Learning

In this section, we describe how to apply Laplacian penalized functional to measure the smoothness of the basis vectors of the face space, which plays the key role in our Spatially Smooth Subspace Learning (SSSL) approach . We begin with a general description of Laplacian smoothing.



(a) Basis vector $\mathbf{a}^T$ ($\mathbf{a} \in \mathbb{R}^9$)

(b) Matrix form of $\mathbf{a}$

(c) Basis vector $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$

(d) $\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T$, Basis of tensor space $\mathcal{R}^3 \otimes \mathcal{R}^3$

Figure 6.1: Take face images of size 3×3. The ordinary vector-based subspace learning algorithms (*e.g.* PCA and LDA) first convert the face images to 9-dimensional vectors and compute the basis vectors (projection functions). The basis vector is also 9-dimensional, as shown in (a). (b) The basis vector can be converted to the matrix form and shown as an image, which was referred as Eigenface (PCA) and Fisherface (LDA). The 9 numbers in the basis vector are independent estimated and there is no spatial relation between them. (c) The tensor-based subspace learning approaches directly take 3×3 face images as input and compute a set of 3-dimensional basis vectors $\mathbf{u}$'s and $\mathbf{v}$'s. (d) Each $\mathbf{u}$ and $\mathbf{v}$ form a basis $\mathbf{u} \otimes \mathbf{v}$ in tensor space which can also be shown as an image. The 9 numbers in the tensor basis only have 6 degrees of freedom and the values in the same row (column) have a common divisor. However, there is no guarantee of the spatial smoothness of the basis function.

6.2.1 Laplacian Smoothing

Let f be a function defined on a region of interest, $\Omega \subset \mathbb{R}^d$. The Laplacian operator $\mathcal{L}$ is defined as follows [53]:

$$\mathcal{L}f(\mathbf{t}) = \sum_{j=1}^{d} \frac{\partial^2 f}{\partial t_j^2} \tag{6.2}$$

The Laplacian penalty functional, denoted by $\mathcal{J}$, is defined by:

$$\mathcal{J}(f) = \int_{\Omega} \left[\mathcal{L}f\right]^2 d\mathbf{t} \tag{6.3}$$

Intuitively, $\mathcal{J}(f)$ measures the smoothness of the function f over the region Ω. In this paper, our primary interest is in image. An image is intrinsically a two-dimensional signal. Therefore, we take d to be 2 in the following.

6.2.2 Discretized Laplacian Smoothing

As we described previously, $m_1 \times m_2$ face images can be represented as vectors in $\mathbb{R}^m, m = m_1 \times m_2$. Let $\mathbf{a}_i \in \mathbb{R}^m$ be the basis vectors (projection functions) obtained by subspace learning algorithms. Without loss of generality, $\mathbf{a}_i$ can also be considered as functions defined on a $m_1 \times m_2$ lattice.

For a face image, the region of interest Ω is a two-dimensional rectangle, which for notational convenience we take to be $[0, 1]^2$. A lattice is defined on Ω as follows. Let $\mathbf{h} = (h_1, h_2)$ where $h_1 = 1/m_1$ and $h_2 = 1/m_2$. Ω_h consists of the set of two-dimensional vectors $\mathbf{t}_i = (t_{i_1}, t_{i_2})$ with $t_{i_j} = (i_j - 0.5) \cdot h_j$ for $1 \leq i_j \leq n_j$ and $1 \leq j \leq 2$. There are a total of $m = m_1 \times m_2$ grid points in this lattice. Let D_j be an $m_j \times m_j$ matrix that yields a discrete approximation to $\partial^2/\partial t_j^2$. Thus if $\mathbf{u} = (u(t_1), \cdots, u(t_{m_j}))$ is an m_j-dimensional

vector which is a discretized version of a function $u(t)$, then D_j has the property that:

$$[D_j \mathbf{u}]_i \approx \frac{\partial^2 u(t_i)}{\partial t^2}$$

for $i = 1, \cdots, m_j$. There are many possible choices of D_j [9]. In this work, we apply the modified Neuman discretization [64]:

$$D_j = \frac{1}{h_j^2} \begin{pmatrix} -1 & 1 & & & & & & 0 \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & . & . & . & & & \\ & & & 1 & -2 & 1 & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 & 1 \\ 0 & & & & & & 1 & -1 \end{pmatrix}$$

Give D_j, a discrete approximation for two-dimensional Laplacian $\mathcal{L}$ is the $m \times m$ matrix:

$$\Delta = D_1 \otimes I_2 + I_1 \otimes D_2 \tag{6.4}$$

where I_j is $m_j \times m_j$ identity matrix for $j = 1, 2$. $\otimes$ is the kronecker product [49].

For a $m_1 \times m_2$ dimensional vector **a**, it is easy to check that $\|\Delta \cdot \mathbf{a}\|^2$ is proportional to the the sum of the squared differences between nearby grid points of **a** with its matrix form. It provides a measure of smoothness of **a** on the $m_1 \times m_2$ lattice.

6.2.3 The Algorithm

Given a pre-defined graph structure with weight matrix W, the SSSL approach is defined as the maximizer of

$$\frac{\mathbf{a}^T X W X^T \mathbf{a}}{(1-\alpha)\mathbf{a}^T X D X^T \mathbf{a} + \alpha \mathcal{J}(\mathbf{a})}, \tag{6.5}$$

where $\mathcal{J}$ is the discretized Laplacian regularization functional:

$$\mathcal{J}(\mathbf{a}) = \|\Delta \cdot \mathbf{a}\|^2 = \mathbf{a}^T \Delta^T \Delta \mathbf{a}. \tag{6.6}$$

The parameter $0 \leq \alpha \leq 1$ controls the smoothness of the estimator.

The vectors $\mathbf{a}_i$ $(i = 1, \cdots, l)$ that maximize the objective function (6.5) are given by the maximum eigenvalue solutions to the following generalized eigenvalue problem.

$$XWX^T\mathbf{a} = \lambda \left((1-\alpha)XDX^T + \alpha\Delta^T\Delta\right)\mathbf{a}. \tag{6.7}$$

With the choices of different W as described in Section 2, our approach gives the spatially smooth version of LDA, LPP and NPE.

6.3 Experimental Results

In this section, several experiments are carried out to show the effectiveness of our proposed Spatially Smooth Subspace Learning (SSSL) approach for face representation and recognition.

6.3.1 Face Representation Using Smooth Fisherfaces

In the last section, we have discussed how to learn a spatially smooth face subspace. The images of faces in the training set are used to learn such a subspace. The subspace is spanned by the eigenvectors corresponding to the largest eigenvalues in Eq. (6.7). We can display the eigenvectors as images.

When we use the spatially smooth LDA approach, these images may be called *Smooth Fisherfaces* (S-Fisherfaces). Using the Yale face database as the training set, we present the first seven S-Fisherfaces in Fig. (6.2), together with Eigenfaces, Fisherfaces and 2DLDA

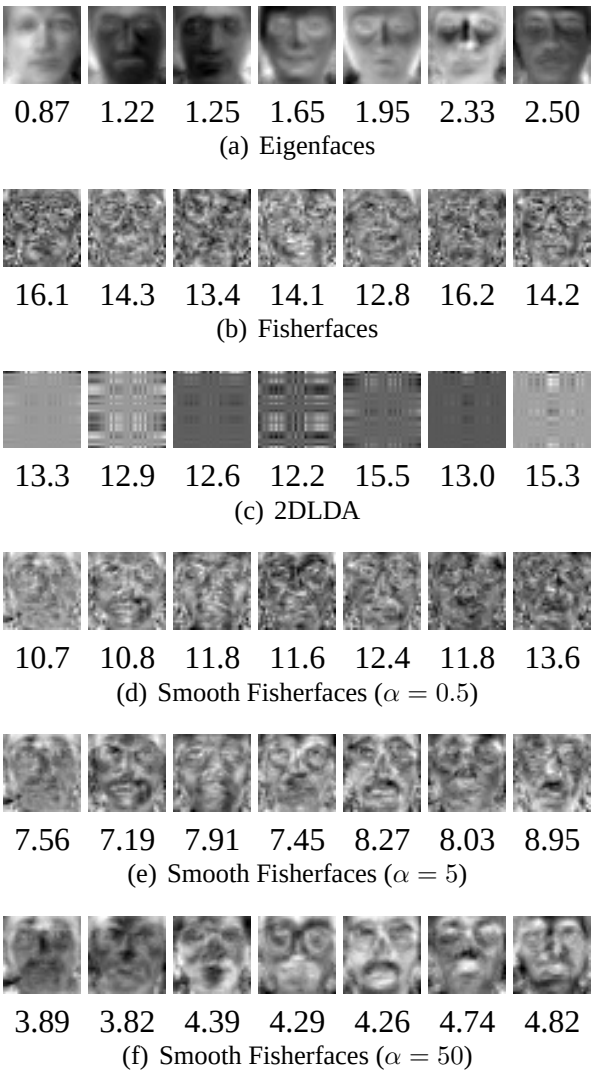


Figure 6.2: (a) $\sim$ (e) The first 7 Eigenfaces, Fisherfaces, and Smooth Fisherfaces calculated from the face images in the Yale database. For each face (eigenvector **a**), we also calculated and showed the $\|\Delta \cdot \mathbf{a}\|$ below of each image. Since each eigenvector is normalized, $\|\Delta \cdot \mathbf{a}\|$ can measure the spatial smoothness of **a**. S-Fisherfaces is smoother than Fisherfaces. With bigger α, S-Fisherfaces become much smoother. (g) The bases of 2DLDA, a tensor extension of LDA. The five bases are $\mathbf{u}_1\mathbf{v}_1^T$, $\mathbf{u}_2\mathbf{v}_1^T$, $\mathbf{u}_1\mathbf{v}_2^T$, $\mathbf{u}_2\mathbf{v}_2^T$, $\mathbf{u}_3\mathbf{v}_1^T$, $\mathbf{u}_1\mathbf{v}_3^T$ and $\mathbf{u}_3\mathbf{v}_3^T$. It is interesting to note that the Eigenfaces are smoothest.

[85]. Note that there is a parameter α which controls the smoothness in S-Fisherfaces. Fig. (6.2) shows three groups S-Fisherfaces with $\alpha = 0.5$, 5 and 50. For each face (eigenvector **a**), we also calculated the $\|\Delta \cdot \mathbf{a}\|$ which can measure the spatial smoothness of **a**.

We can see that S-Fisherfaces is smoother than Fisherfaces. The bigger α is, the smoother are S-Fisherfaces. It is interesting to note that the Eigenfaces are smoothest. However, Eigenfaces do not encode discriminating information thus are not optimal for recognition. As we discussed in Section 6.1, the bases of tensor approaches only consider the relationship of pixels in the same row (or column), thus the bases in 2DLDA are still spatially rough. S-Laplacianfaces consider both the discriminating power and the spatial correlation between the pixels in the face images.

6.3.2 Face Recognition Using SSSL Approach

In this section, we investigate the performance of our proposed Spatially Smooth Subspace Learning approach for face recognition. The face recognition task is handled as a multi-class classification problem − we map each test image to a low-dimensional subspace via the embedding learned from training data, and then classify the test data by the nearest neighbor criterion.

Datasets and Compared Algorithms

The Yale and AT&T face databases are used in our experiments. The Yale face database[1] contains 165 gray scale images of 15 individuals, each individual has 11 images. The images demonstrate variations in lighting condition, facial expression (normal, happy, sad, sleepy, surprised, and wink).

The AT&T face database[2] consists of a total of 400 face images, of a total of 40 people (10 samples per person). The images were captured at different times and have different

[1] http://cvc.yale.edu/projects/yalefaces/yalefaces.html
[2] http://www.cl.cam.ac.uk/Research/DTG/attarchive/facesataglance.html

Table 6.1: Compared algorithms

Objective function	Ordinary version	Tensor extension	Smooth version
PCA	Eigenface [78]	CSA [82]	–
LDA	Fisherface [3]	2DLDA [85]	S-LDA
LPP	Laplacianface [47]	TSA [44]	S-LPP
NPE	NPE [45]	TNPE	S-NPE
MFA (LDE)	MFA [84]	TMFA [84]	S-MFA

variations including expressions (open or closed eyes, smiling or non-smiling) and facial details (glasses or no glasses). The images were taken with a tolerance for some tilting and rotation of the face up to 20 degrees.

All the face images are manually aligned and cropped. The size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. The features (pixel values) are then scaled to [0,1] (divided by 256). For the vector-based approaches, the image is represented as a 1024-dimensional vector, while for the tensor-based approaches the image is represented as a (32×32)-dimensional matrix, or the second order tensor.

The image set is then partitioned into the gallery and probe set with different numbers. For ease of representation, Gm/Pn means m images per person are randomly selected for training and the remaining n images are for testing.

Table 6.1 summarizes the 14 algorithms compared in our experiments. These algorithms belong to five families, *i.e.*, PCA family, LDA family, LPP [46] family, NPE [45] family and MFA [84] (LDE [25]) family. For each family, we take the ordinary vector-based approach, *i.e.*, Eigenface [78], Fisherface [3], Laplacianface [47], NPE [45] and MFA [84]. We also take their tensor extensions (or 2D extensions), *i.e.* CSA [82], 2DLDA [85], TSA [44], TNPE and TMFA [84] respectively. Finally, we implement their spatially smooth versions by using 2-D Laplacian smoothing regularization technique, which leads to S-LDA, S-LPP, S-NPE and S-MFA.

Table 6.2: Recognition accuracy on Yale database (mean±std-dev%)

Method	G2/P9	G3/P8	G4/P7	G5/P6
Eigenface	46.0±3.4	50.0±3.5	55.7±3.5	57.7±3.8
CSA	49.4±3.5	54.0±3.0	57.8±3.3	59.8±3.9
Fisherface	45.7±4.2	62.3±4.5	73.0±5.4	76.9±3.2
2DLDA	43.4±6.2	56.3±4.7	63.5±5.6	66.1±4.8
S-LDA	**57.6±4.1**	**72.3±4.4**	**77.8±3.0**	**81.7±3.2**
Laplacianface	54.5±5.2	67.2±4.1	72.7±4.2	75.8±4.6
TSA	44.3±6.5	55.8±4.5	63.2±6.0	65.7±4.6
S-LPP	**57.9±4.5**	**72.0±4.0**	**76.0±3.4**	**81.4±2.9**
NPE	52.6±4.0	66.0±4.6	73.2±5.0	76.4±4.4
TNPE	43.4±6.2	56.8±3.9	61.8±3.5	63.0±3.4
S-NPE	**57.5±4.7**	**71.9±3.9**	**77.0±3.4**	**80.9±3.5**
MFA	45.7±4.2	62.3±4.5	73.0±5.4	76.9±3.2
TMFA	43.4±6.2	56.3±4.7	63.5±5.6	66.1±4.8
S-MFA	**57.2±4.3**	**71.2±4.0**	**76.9±3.1**	**81.1±3.1**

Face recognition results

The recognition accuracy of different algorithms on Yale and AT&T databases are reported on the Table (6.2) and (6.3) respectively. For each given l (the number of training images per individual), we average the results over 20 random splits and report the mean as well as the standard deviation. The cross validation in the training set was used to select the parameter α in those SSSL approaches (S-LDA, S-LPP, S-NPE and S-MFA).

A crucial problems for most of the subspace learning based face recognition methods is dimensionality estimation. The performance usually varies with the number of dimensions. We show the best results obtained by those ordinary subspace learning algorithms and their tensor extensions. Since the cross validation is needed to estimate the parameter α for those SSSL approaches, we simply set the dimensionality as $c - 1$ for those SSSL approaches where c is the number of individuals.

The main observations from the performance comparisons include:

- SSSL approach significantly outperforms the corresponding ordinary subspace learning algorithm and the tensor extension with different numbers of training samples per individual in both the two databases. The reason lies SSSL explicitly takes into

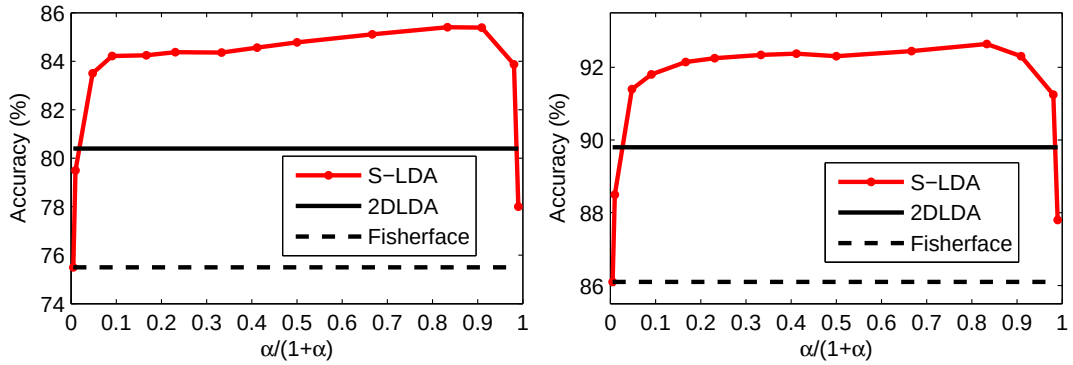
Table 6.3: Recognition accuracy on AT&T database (mean±std-dev%)

Method	G2/P8	G3/P7	G4/P6	G5/P5
Eigenface	70.7±2.7	78.9±2.3	84.2±2.1	87.9±2.5
CSA	71.3±2.6	79.9±2.2	84.8±1.9	88.1±2.5
Fisherface	75.5±3.3	86.1±1.9	91.6±1.9	94.3±1.4
2DLDA	80.4±3.0	89.8±2.1	93.5±1.7	95.8±1.2
S-LDA	**85.2±2.2**	**92.3±1.7**	**95.8±1.3**	**97.2±1.3**
Laplacianface	77.6±2.5	86.0±2.0	90.3±1.7	93.0±1.9
TSA	80.4±3.2	89.8±2.1	93.4±1.6	95.7±1.3
S-LPP	**85.2±2.2**	**92.3±1.7**	**95.8±1.3**	**97.2±1.3**
NPE	77.6±2.7	85.7±1.8	90.5±1.8	93.4±1.8
TNPE	80.4±3.0	87.6±2.2	91.5±1.7	93.7±2.3
S-NPE	**84.8±2.3**	**92.3±1.7**	**95.4±1.2**	**96.9±0.9**
MFA	75.4±3.1	86.1±1.9	91.6±1.9	94.3±1.4
TMFA	80.4±3.0	89.8±2.1	93.7±1.7	95.8±1.2
S-MFA	**84.9±2.3**	**92.4±1.3**	**95.8±1.5**	**97.4±1.2**

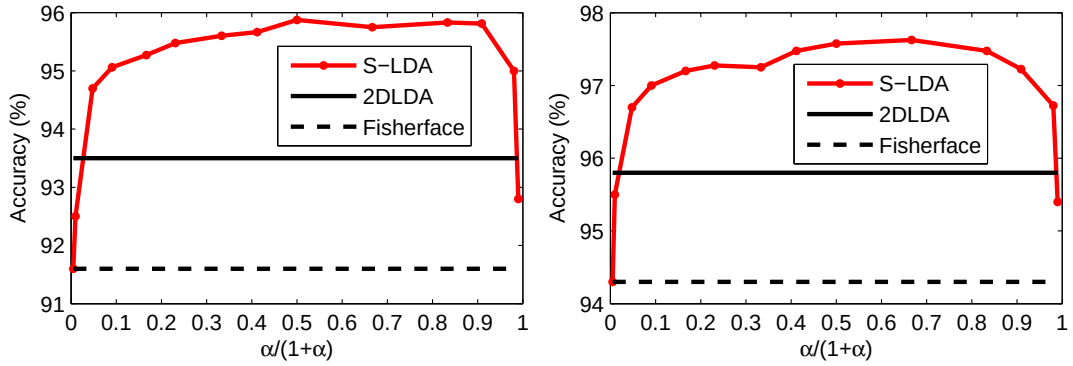
account the spatial relationship between the pixels in an image. The use of spatial information significantly reduces the number of degrees of freedom. Therefore, SSSL can have good performance even when there is only a small number of training samples available.

- The methods based on PCA (Eigenface and CSA) perform the worst in most the cases. This is probably due to the fact that the PCA is unsupervised and does not encode discriminating information.

- The tensor-based algorithms show their advantages on AT&T database while failed gain improvement on Yale database. This suggests that the spatial relationship of face images considered in tensor-based approach (relation between the pixels in the same row or column) has its limitation. Compare to the tensor approaches, our SSSL approach is a more natural extension of incorporating spatial information in vector-based algorithm, which is supported by the experimental results.



(a) 2 Train (b) 3 Train



(c) 4 Train (d) 5 Train

Figure 6.3: Model selection for S-LDA on AT&T database. The curve shows the accuracy of S-LDA with respect to $\alpha/(1+\alpha)$. The solid line shows the accuracy of 2DLDA and the dashed line shows the performance of Fisherface.

Model selection for SSSL

The $\alpha \geq 0$ is an essential parameter in our SSSL approaches which controls the smoothness of the estimator. We use cross validation on the training set to select this parameter in the previous experiments. In this subsection, we take S-LDA as an example to study the impact of parameter α on the recognition performance.

Figure (6.3) shows the performance of S-LDA as a function of the parameter α on AT&T database. For convenience, the X-axis is plotted as $\alpha/(1+\alpha)$ which is strictly in the interval $[0, 1]$. Each figure has three lines. The curve shows the accuracy of S-LDA with respect to $\alpha/(1+\alpha)$. The solid line shows the accuracy of 2DLDA and the dashed line shows the performance of Fisherface. It is easy to see that S-LDA can achieve significantly better performance than both 2DLDA and Fisherface over a large range of α. Thus, the

parameter selection is not a very crucial problem in S-LDA algorithm.

Chapter 7

Conclusions

In this thesis, we propose a new dimensionality reduction algorithm called *Spectral Regression* (SR). It is based on the same variational principle that gives rise to the Laplacian Eigenmap [4]. As a natural extension of several recent nonparametric techniques for global nonlinear dimensionality reduction such as [68, 75, 4], SR aims at learning an embedding function (either linear or in RKHS) which is defined everywhere (and therefore on novel test data points). It casts the problem of learning an embedding function into a regression framework which facilitates both efficient computation and the use of regularization techniques. The computational complexity analysis illustrates the advantage of SR over other linear or kernel extensions of LLE and Laplacian Eigenmap [46, 7, 45].

By using the affinity graph to model both label and local neighborhood information, SR can make efficient use of both labeled and unlabeled points to discover the intrinsic discriminant structure in the data. Our theoretical analysis linked our algorithm to LDA [40] and LPP [46] in supervised and unsupervised cases. The experimental results on classification and semi-supervised classification demonstrate the effectiveness and efficiency of our algorithm.

Our approach provides a general framework for learning a function (either linear or in RKHS) in graph embedding approaches. With the specific affinity graph, SR can provide a natural out-of-sample extension of many spectral embedding algorithms like LLE, Isomap, Laplacian Eigenmaps and spectral clustering algorithms [62].

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[2] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, (12):2385–2404, 2000.

[3] P. N. Belhumeur, J. P. Hepanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, Cambridge, MA, 2001.

[5] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[6] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2003.

[7] M. Brand. Continuous nonlinear dimensionality reduction by kernel eigenmaps. In *International Joint Conference on Artificial Intelligence*, 2003.

[8] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[9] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving poisson's equations. *SIAM Journal on Numerical Analysis*, 7(4):627–656, Dec. 1970.

[10] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, December 2005.

[11] D. Cai, X. He, and J. Han. Subspace learning based on tensor analysis. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2005-2572, May 2005.

[12] D. Cai, X. He, and J. Han. Efficient kernel discriminant analysis via spectral regression. In *Proc. Int. Conf. on Data Mining (ICDM'07)*, 2007.

[13] D. Cai, X. He, and J. Han. Isometric projection. In *Proc. 2007 AAAI Conf. on Artificial Intelligence (AAAI-07)*, 2007.

[14] D. Cai, X. He, and J. Han. Spectral regression: A unified approach for sparse subspace learning. In *Proc. Int. Conf. on Data Mining (ICDM'07)*, 2007.

[15] D. Cai, X. He, and J. Han. Spectral regression: A unified subspace learning framework for content-based image retrieval. In *Proceedings of the ACM Conference on Multimedia*, 2007.

[16] D. Cai, X. He, and J. Han. Spectral regression for dimensionality reduction. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2856, May 2007.

[17] D. Cai, X. He, and J. Han. Spectral regression for efficient regularized subspace learning. In *Proc. Int. Conf. Computer Vision (ICCV'07)*, 2007.

[18] D. Cai, X. He, and J. Han. SRDA: An efficient algorithm for large scale discriminant analysis. Technical report, Computer Science Department, UIUC, UIUCDCS-R-2007-2857, May 2007.

[19] D. Cai, X. He, and J. Han. SRDA: An efficient algorithm for large scale discriminant analysis. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):1–12, January 2008.

[20] D. Cai, X. He, W. V. Zhang, and J. Han. Regularized locality preserving indexing via spectral regression. In *Proc. 2007 ACM Int. Conf. on Information and Knowledge Management (CIKM'07)*, 2007.

[21] D. Cai, X. He, K. Zhou, J. Han, and H. Bao. Locality sensitive discriminant analysis. In *International Joint Conference on Artificial Intelligence*, 2007.

[22] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[23] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[24] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 16*, 2003.

[25] H.-T. Chen, H.-W. Chang, and T.-L. Liu. Local discriminant embedding and its variants. In *Proc. 2005 Internal Conference on Computer Vision and Pattern Recognition*, 2005.

[26] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2nd edition, 2001.

[28] A. d'Aspremont, L. E. Chaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. In *Advances in Neural Information Processing Systems 17*, 2004.

[29] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[30] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Jounal of Artificial Intelligence Research*, (2):263–286, 1995.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[32] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[33] M. A. Fanty and R. Cole. Spoken letter recognition. In *Advances in Neural Information Processing Systems 3*, 1990.

[34] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[35] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[36] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

[37] A. Grama, G. Karypis, V. Kumar, and A. Gupta. *An Introduction to Parallel Computing: Design and Analysis of Algorithms*. Addison Wesley, 2nd edition, 2003.

[38] S. Guattery and G. L. Miller. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21(3):703–723, 2000.

[39] J. Ham, D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. 2004.

[40] T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.

[41] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[42] X. He. Incremental semi-supervised subspace learning for image retrieval. In *Proceedings of the ACM Conference on Multimedia*, New York, October 2004.

[43] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality preserving indexing for document representation. In *Proc. 2004 Int. Conf. on Research and Development in Information Retrieval (SIGIR'04)*, pages 96–103, Sheffield, UK, July 2004.

[44] X. He, D. Cai, and P. Niyogi. Tensor subspace analysis. In *Advances in Neural Information Processing Systems 18*, 2005.

[45] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In *Proc. Int. Conf. Computer Vision (ICCV'05)*, 2005.

[46] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2003.

[47] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[48] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. 1999 Int. Conf. on Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, Berkeley, CA, Aug. 1999.

[49] R. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[50] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. pages 762–768, 1997.

[51] D. P. Huijsmans and N. Sebe. How to complete performance graphs in content-based image retrieval: Add generality and normalize scope. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):245–251, 2005.

[52] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5), 1994.

[53] J. Jost. *Riemannian Geometry and Geometric Analysis*. Springer-Verlag, 2002.

[54] J. M. Lee. *Introduction to Smooth Manifolds*. Springer-Verlag New York, 2002.

[55] Y.-Y. Lin, T.-L. Liu, and H.-T. Chen. Semantic manifold learning for image retrieval. In *Proceedings of the ACM Conference on Multimedia*, Singapore, November 2005.

[56] L. Lovasz and M. Plummer. *Matching Theory*. Akadémiai Kiadó, North Holland, Budapest, 1986.

[57] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1980.

[58] C. A. Micchelli. Algebraic aspects of interpolation. In *Proceedings of Symposia in Applied Mathematics*, volume 36, pages 81–102, 1986.

[59] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Proc. of IEEE Neural Networks for Signal Processing Workshop (NNSP)*, 1999.

[60] B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In *Advances in Neural Information Processing Systems 18*, 2005.

[61] B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 641–648, 2006.

[62] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, Cambridge, MA, 2001.

[63] C. L. Novak and S. A. Shafer. Anatomy of a color histogram. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'92)*, pages 599–605, 1992.

[64] F. O'Sullivan. Discretized laplacian smoothing by fourier methods. *Journal of the American Statistical Association*, 86(415):634–642, Sep. 1991.

[65] C. C. Paige and M. A. Saunders. Algorithm 583 LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software*, 8(2):195–209, June 1982.

[66] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, March 1982.

[67] R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.

[68] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[69] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interative content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5), 1998.

[70] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[71] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proc. 2005 Int. Conf. Machine Learning (ICML'05)*, 2005.

[72] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.

[73] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, 2001.

[74] M. A. Stricker and M. Orengo. Similarity of color images. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 381–392, 1995.

[75] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[76] A. N. Tikhonov. Regularization of incorrectly posed problems. *Soviet Math.*, (4), 1963 (English Translation).

[77] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001.

[78] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[79] V. N. Vapnik. *Statistical learning theory*. John Wiley & Sons, 1998.

[80] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[81] T. Xiong, J. Ye, Q. Li, V. Cherkassky, and R. Janardan. Efficient kernel discriminant analysis via QR decomposition. In *Advances in Neural Information Processing Systems 17*, 2004.

[82] D. Xu, S. Yan, L. Zhang, H.-J. Zhang, Z. Liu, and H.-Y. Shum. Concurrent subspace analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[83] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proc. 2003 Int. Conf. on Research and Development in Information Retrieval (SIGIR'03)*, pages 267–273, Toronto, Canada, Aug. 2003.

[84] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extension: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), 2007.

[85] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *Advances in Neural Information Processing Systems 17*, 2004.

[86] J. Yu and Q. Tian. Learning image manifolds by semantic subspace projection. In *Proceedings of the ACM Conference on Multimedia*, Santa Barbara, October 2006.

[87] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2003.

[88] H. Zhou and T. Hastie. Regression shrinkage and selection via the elastic net, with applications to microarrays. Technical report, Statistics Department, Stanford University, 2003.

[89] H. Zhou, T. Hastie, and R. Tibshirani. Sparse principle component analysis. Technical report, Statistics Department, Stanford University, 2004.

Author's Biography

Deng Cai was born on February 25th, 1978 in People's Republic of China. He attended Tsinghua University, China where he earned both the bachelor and master of engineering in automation in 2000 and 2003, respectively. He received his Ph.D. from the University of Illinois at Urbana-Champaign in 2009 with her dissertation entitled *Spectral Regression: A Regression Framework for Efficient Regularized Subspace Learning*.

The dissertation was completed and defended on January 30th, 2009. It was submitted to the graduate school in April 2009.